Draft Indian Standard

Smart Cities - Data Exchange Framework: Part 1 Reference Architecture

(First Revision)

BUREAU OF INDIAN STANDARDS

MANAK BHAVAN, 9 BAHADUR SHAH ZAFAR MARG

NEW DELHI 110002

May 2019

CONTENTS

FOREWORD

This is a First Revision of the preliminary draft of the standard reference architecture for data exchange, under review at the Bureau of Indian Standards, Smart Infrastructure Sectional Committee LITD28.

This standard has been requested by the Ministry of Housing Affairs, Government of India.

INTRODUCTION

The next phase of smart cities implementations will leverage **data empowerment**, in order to harness the maximum value from the enormous data cities generate. The current smart city implementations are unable to satisfy this need efficiently, due to the proprietary and ad-hoc nature of the interfaces and their implementations. Hence it is difficult to develop next generation AI/ML based applications for providing new solutions and services at scale, in the current framework. The Data Exchange Framework as discussed in this document aims to address this gap, **by creating a reference architecture (part 1) and interface specifications (part 2)** for interconnecting various IT systems of different government departments as well as external organizations**.**

The data exchange will provide two key services:

- A **catalogue service** which will host a catalogue of meta-information about the various data sets, with information about the custodian of the data, data model for the data, the API endpoints, API methods etc.
- An **authorization service** that will enable a data custodian (one who is responsible for the data) to regulate access to their data sets.

Security and Privacy will be incorporated by design in this architecture. This framework should simplify the life of the data custodian as well as the application developer.

The data exchange framework will enable new applications to emerge, that can take advantage of data from different IT Systems, to provide novel services. For example, a Woman's safety index can calculate the live safety index of any street, combining data from smart streetlights, video analytics from traffic cameras, data from police database along with analysis of land use. Such an index can be used by trip planning apps to allow for determining safe routes or used by city or police to plan on patrolling.

By defining the reference architecture and specifying the interfaces and data models, the data exchange framework standards will enable a whole new ecosystem of application developers to provide new, data driven, solutions and services. Additionally, adopting the data exchange framework nationally, will enable economies of scale for the developers and will allow same applications to run across the country. For data custodians – the data exchange framework will allow a simple way to expose, give consent, audit and track their data usage.

1. SCOPE

This document describes the reference architecture for the data exchange framework, the use cases that are enabled in this ecosystem and the responsibilities of the various stakeholders, their interactions with other system stakeholders.

It describes the high level architecture of the three main components of the data exchange services:

a) Catalogue service that provides framework to manage meta-information about resources,

b) Authorisation service, that manages authorisation to access the resources

c) Resource Access Service, that provides a standardized way to access resources.

The current document also describes high level definition of the various interfaces. A more detailed specification for the data exchange framework is described in part 2.

2. REFERENCES

2.1. Normative References

The following referenced documents are necessary for the application of the present document.

[1] IETF RFC 7231: "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content". Available at https://tools.ietf.org/html/rfc7231.

[2] IETF RFC 7232: "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests". Available at https://tools.ietf.org/html/rfc7232.

[3] IETF RFC 3986: "Uniform Resource Identifier (URI): Generic Syntax". Available at https://tools.ietf.org/html/rfc3986.

[4] IETF RFC 8259: "The JavaScript Object Notation (JSON) Data Interchange Format". Available at https://tools.ietf.org/html/rfc8259.

[5] IETF RFC 8288: "Web Linking". Available at https://tools.ietf.org/html/rfc8288.

[6] IETF RFC 7946: "The GeoJSON Format". Available at https://tools.ietf.org/html/rfc7946.

[7] IETF RFC 8141: "Uniform Resource Names (URNs)". Available at https://tools.ietf.org/html/rfc8141.

[8] Open Geospatial Consortium Inc. OGC 06-103r4: "OpenGIS® Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture". Available at https://portal.opengeospatial.org/files/?artifact_id=25355.

[9] UN/CEFACT Common Codes for specifying the unit of measurement. Available at http://www.unece.org/fileadmin/DAM/cefact/recommendations/rec20/rec20_Rev9e_2014.xls.

[10] IETF RFC 7396: "JSON Merge Patch". Available at https://tools.ietf.org/html/rfc7396.

[11] ISO 8601: 2004: "Data elements and interchange formats -- Information interchange -- Representation of dates and times". Available at http://www.iso.org/iso/catalogue_detail?csnumber=40874.

[12] IETF RFC 2818: "HTTP Over TLS". Available at https://tools.ietf.org/html/rfc2818.

[13] IETF RFC 5246: "The Transport Layer Security (TLS) Protocol Version 1.2". Available at https://tools.ietf.org/html/rfc5246.

[14] IANA Registry of Link Relation Types. Available at https://www.iana.org/assignments/link-relations/.

[15] ISO/IEC 29100:2011(en) Information technology — Security techniques — Privacy framework. Available at https://www.iso.org/obp/ui/#iso:std:iso-iec:29100:ed-1:v1:en

[16] The OAuth 2.0 Authorization Framework. Available at https://tools.ietf.org/html/rfc6749

[17] OAuth 2.0 Token Revocation. Available at https://tools.ietf.org/html/rfc7009

[18] MQTT 5.0, OASIS Standard. Available at https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html

[19] ISO/IEC 19464: Information technology — Advanced Message Queuing Protocol (AMQP) v1.0 specification. Available at https://standards.iso.org/ittf/PubliclyAvailableStandards/c064955_ISO_IEC_19464_2014.zip

[20] IETF RFC 6455, The WebSocket Protocol. Available at http://www.ietf.org/rfc/rfc6455.txt

[21] IETF RFC 2326, Real Time Streaming Protocol (RTSP). Available at http://www.ietf.org/rfc/rfc2326.txt

[22] ISO 19119:2016 Geographic Information - Services. Available at https://www.iso.org/standard/59221.html

2.3. Informative References

[i.1] The Personal Data Protection Bill 2018, Govt. of India, http://meity.gov.in/writereaddata/files/Personal_Data_Protection_Bill,2018.pdf

[i.2] Electronic Consent Framework, Technical Specs v1.1, http://dla.gov.in/sites/default/files/pdf/MeitY-Consent-Tech-Framework%20v1.1.pdf

[i.3] National Data Sharing and Accessibility Policy 2012, Govt. of India, https://data.gov.in/sites/default/files/NDSAP.pdf

[i.4] Account Aggregator Technical Standards, Version 1.2, Reserve Bank Information Technology Pvt. Ltd., https://api.rebit.org.in/group

[i.5] Policy on Open Programming Interfaces of Govt. of India, 2015. http://meity.gov.in/writereaddata/files/Open_APIs_19May2015.pdf

[i.6] User-Managed Access (UMA) 2.0, https://docs.kantarainitiative.org/uma/ed/uma-core-2.0-08.html

[i.7] User-Managed Access (UMA) 2.0 Grant for OAuth 2.0 Authorization. Available at https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-grant-2.0.html

[i.8] Federated Authorization for User-Managed Access (UMA) 2.0. Available at https://docs.kantarainitiative.org/uma/wg/rec-oauth-uma-federated-authz-2.0.html

[i.9] ETSI GS CIM 009 V1.1.1 (2019-01), Context Information Management (CIM); NGSI-LD API. Available at https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.01.01_60/gs_CIM009v010101p.pdf

[i.10] ONVIF Network Interface Specifications. Available at https://www.onvif.org/profiles/specifications/

[i.11] HTTP Live Streaming. Available at https://tools.ietf.org/html/draft-pantos-http-live-streaming-23

[i.12] PAS 212:2016 Automatic resource discovery for the internet of things. Specification. Available at https://shop.bsigroup.com/forms/PASs/PAS-212-2016-download/

[i.13] ISO/IEC 27001, Information technology – Security techniques – Information security management systems – Requirements

[i.14] ISO/IEC 27002, Information technology – Security techniques – Code of practice for information security controls

[i.15] ISO/IEC 27017, Information technology – Security techniques – Code of practice for information security controls based on ISO/IEC 27002 for cloud services

[i.16] ISO/IEC 27018, Information technology – Security techniques – Code of practice for protection of personally identifiable information (PII) in public clouds acting as PII processors

[i.17] ISO/IEC 27031, Information technology – Security techniques – Guidelines for information and communication technology readiness for business continuity

[i.18] ISO/IEC 27033 (all parts), Information technology – Security techniques – Network security

[i.19] ISO/IEC 27034 (all parts), Information technology – Security techniques – Application security

[i.20] ISO/IEC 27035 (all parts), Information technology – Security techniques – Information security incident management

[i.21] ISO/IEC 27040, Information technology – Security techniques – Storage security
ISO/IEC 29100, Information technology – Security techniques – Privacy framework

[i.22] ISO/IEC29101, Information technology – Security techniques – Privacy architecture framework

[i.23] ISO/IEC 29134:2017, Information technology – Security techniques – Guidelines for privacy impact assessment

[i.24] ISO/IEC 29151, Information technology – Security techniques – Code of practice for personally identifiable information protection

3. TERMINOLOGY AND DEFINITIONS

3.1. Conventions used in the Document

- The key terminologies use consolas font type and dark cornflower blue 3 color encoding.
- The lowerCamelCase is used in attribute naming. For nouns, UpperCamelCase is used.

Table 3-1: Terminology

Term	**Explanation**
Provider	**Legal Entity**: Human (possibly delegated by an Organization), Organization or an organizational role that has responsibility to provide authorisation to use resources.
Resource Server	**Service**: Serves resources to authorized Apps/Consumers.
Consumer	**Legal Entity**: Human or Organization or an organizational Role that consumes a resource via a web or mobile App.
App	**Application:** Software (like a mobile app, web app, device app or server app), that uses resources to provide a service or experience to the Consumer.
ProviderApp	**Application:** An App that enables a Provider to manage the meta-data and access control in the data exchange, for the resources they are responsible for.
Data Exchange Framework	**Service**: Hosts and manages meta-data about resources and manages authorisation for accessing the resources.
Consent	Provider's freely given, specific and informed agreement to the accessing and processing of specific resources in their responsibility.
Consent Artefact	A machine-readable electronic document that specifies the parameters and scope of data sharing that a Provider consents to in any data sharing transaction.

Personally Identifiable Information	Any information that (a) can be used to identify the PII principal to whom such information relates, or (b) is or might be directly or indirectly linked to a PII principal Note 1 to entry: To determine whether a PII principal is identifiable, account should be taken of all the means which can reasonably be used by the privacy stakeholder holding the data, or by any other party, to identify that natural person.
PII Principal	Natural person to whom the personally identifiable information (PII) relates Note 1 to entry: Depending on the jurisdiction and the particular data protection and privacy legislation, the synonym "data subject" can also be used instead of the term "PII principal".
Authorization Token	A digital entity that is used to present the authorization credentials to the Resource Server.
Catalogue	A registry of meta-data about the resources in the data exchange available for consumption
resource-item	An entry in the Catalogue that describes the meta-information of the resource that is hosted in an associated Resource Server
DX Authorization Service	Authorization Service of the data exchange
DX Catalogue Service	Catalogue Service of the data exchange
DX Adapter	Adapter service in front of a non-DX compliant Resource Server
DX Administrator	**Legal Entity**: Responsible for administering, managing and running the data exchange
DX Certificate Authority	**Service:** Certificate Authority service run by the DX

Table 3-2: Abbreviations

Abbreviation	**Definition**
DX	Data Exchange
XML	eXtensible Markup Language

`JSON`	Java Script Object Notation
`API`	Application Programming Interface
`PII`	Personally Identifiable Information
`CA`	Certificate Authority
`RS`	Resource Server
`AS`	Authorization Service
`TLS`	Transport Level Security
`CSR`	Certificate Signing Request
`CCA`	Controller of Certifying Authorities

4. DATA EXCHANGE REFERENCE ARCHITECTURE

The `Data Exchange Framework` is a set of services that enables consumption of resources (like data) by a `Consumer` from one or more `Resource Server`s, based on explicit `Consent` obtained from the `Provider` of the resources.

4.1. System Design Guidelines

The implementation guide adheres to the following guidelines:

1. **Technology Agnostic**: The proposed design in the document is agnostic to applications, programming languages, and platforms and aims at seamless and secure flow of electronic data across different stakeholders.
2. **Reliability and Scaling**: Non-repudiation, consent, digital signatures, logging requirements are used to bolster reliability and accountability of the ecosystem. Asynchronous mechanisms and callbacks are used to improve scalability of the system.
3. **Privacy by Design**: The data exchange implementation defines the data sharing mechanisms, based on Electronic Consent and results in a non-repudiable audit trail. The `Provider` of the data resources controls the consent to access the resources. Hence the `Provider` must ensure that `PII` of `PII principal`s are dealt with as per the laws of the land [i.1,i.2]. The data exchange will also maintain the privacy of consumers and all other actors who interact with the exchange.
4. **Security by Design**: The software and systems must be designed from the ground up to be secure. There must be end-to-end security of data (PKI, Digital Signature Certificates, tamper detection) and it must be network agnostic and data centric.
5. **Minimalist and Evolutionary Design**: The design should be simple and minimalistic. It should not present adoption barriers for the ecosystem. The design of the systems should be evolutionary - their capabilities should be built incrementally while allowing for rapid adoption.
6. **Consumer Centric**: The `Consumer` experience and ease of use are critical to successfully deliver various services in the ecosystem. The design principles should take into account the various stakeholder responsibilities and mechanisms to simplify interactions and ease the access to authorised data and services.
7. **Consent Driven**: Mechanisms for dynamic discovery, empowerment of the `Provider` in accordance with the consent architecture proposed in [i.2] shall be incorporated to enhance trust and ensure data privacy.
8. **Open APIs for interoperability and Layered Innovation**: People and systems should have programmatic interfaces for sharing and accessing the digital resources available to them. The specification defines the standard APIs to promote interoperability and deliver services that are designed to work with any device, any form factor, and any network [i.3] .
9. **Ecosystem Driven Approach**: An ecosystem approach, encouraging 3rd party `App Developers`, needs the interfaces between the `Consumers`, `Resource Servers` and

other related services be well defined and standardized. Hence, there must exist a standardized technology backbone that would hold together this ecosystem.

10. **Transparency and Accountability through Data** : Public Open Data [i.3] shall be made available via APIs for transparency. The access to such an open data will ensure high-quality analytics, accurate fraud detection, shorter cycles for system improvement and, most importantly, high responsiveness to consumer's needs.

4.2. Entities Roles and Responsibilities

The following table outlines the roles and responsibilities of the various entities involved in the data exchange ecosystem.

Table 4-1: Entities and their roles

Entity	**Roles and Responsibilities**
`Provider`	**Legal Entity**: Human or Organization that owns a resource.
`Consumer`	**Legal Entity**: Human or Organization that consumes a resource.
`Data Exchange`	**Service**: Hosts and manages meta-data about resources and manages authorisation for those.
`Resource Server`	**Service**: Serves Provider's resources to authorized clients.
`App`	**Application**: A Consumer's application, that consumes resources. Can be a mobile app, web app or server app.
`Identity Service`	**Service**: Provides identities for various entities
`Certificate Authority`	**Service:** Provides Digital Certificates. Their trust can be traced to the Root Certificate Authorities.
`Provider App`	**Application**: Helper app used by Providers to manage interactions with the data exchange. Can be a mobile app, web app or server app.
`App Developer`	**Legal Entity**: Develops Apps that consume (or produce) resources.
`Data Exchange Provider`	**Legal Entity**: Operates the data exchange.

4.3. High Level Architecture

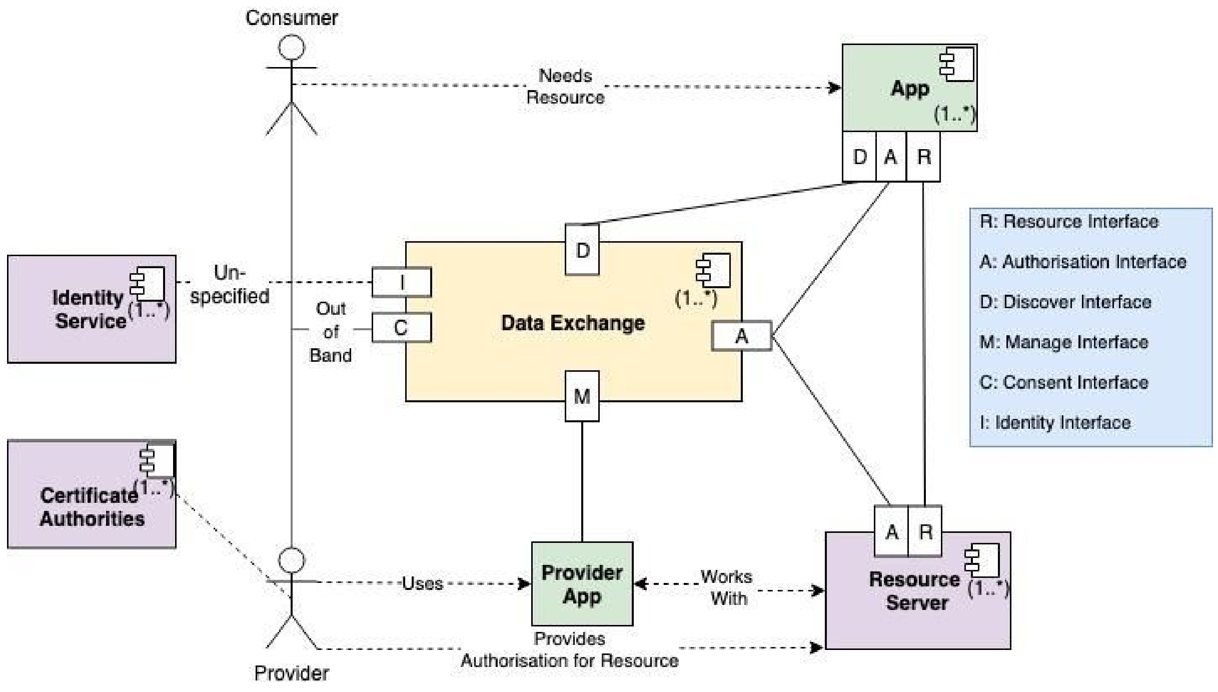

Figure 4-1: Data Exchange Architecture

Resources, managed by a Provider, are hosted on one or more Resource Servers, and are made available for consumption to entities via a description of its meta-information (like its format, Provider, etc.), through a catalogue in the Data Exchange. The Catalogue is both human readable as well as machine-readable.

The Provider registers and manages the meta-data of its resources and their associated access control policies via the management interface of the data exchange. The Provider may use a helper application, like the ProviderApp. The meta-data of each resource should help an app developer to ease the consumption of resources in order to create useful applications for Consumers.

The App can register with the Data Exchange to get notified about any changes to the meta-data of the resources of interest to the Consumer. The App obtains consent to consume the resources via the authorisation interface by obtaining an Authorization Token.

Any request to a provider's resource by a Consumer App will be checked against the existing access control policies. If no decision can be made, the Data Exchange will coordinate between the Provider and the Consumer to complete a consent transaction and generate the Consent Artefact. The Consent Artefact will be used to update the access control policy for those resources. The Consent Artefact will be logged by the system to ensure auditability of the consent flows. The coordination between the Provider and Consumer is done outside the scope of this specification and can be built using any of the available messaging technologies like SMS/OTP/EMAIL etc. The data licensing terms and conditions will also be outside the scope of this specification. However reference to license may be provided in the meta-data for the resource.

In order to provide for a better consumer experience, the `App Developer` may also enter into a resource licensing agreement with the `Provider`. In this case, the `Consumer` can be shielded from having to get consent separately, as long as the `App Developer` and the `App` adhere to the licensing terms and conditions of the `Provider`.

The set of interfaces for this data exchange framework are listed below:

Table 4-2: Data Exchange Interfaces and functionalities

Interface	**Functionality**	**Remarks**

`Manage`	`Create, Update, Delete, List, Search and View the items in the catalogue.`
	
	`Create, Update, Delete and View the access control policies.`
	
	`List and View information about consumers.`
		Interface for the `Provider` to manage the resource meta-info and access-control policies.

`Discover`	`List, Search, View and Count items in the catalogue`	Search can use complex queries and filters involving Geo, Time and other Attributes
`Authorisation`	`Request, Grant, Revoke, Introspect Access Tokens for resources.`	Federated, OAuth style authorization flows.

`Resource`	`Get Latest Data, Search for resources, Get Status, Get Counts, Subscribe, Update Subscription, Unsubscribe.`
	
	`Playback of live and archived media streams, download media files. Stop, Pause and Stop playback.`
	
	`GIS resources access.`
	
	`Service resources access.`
		Retrieve latest data and search for resources using complex queries and filters. Search can be on Geo, Time and other Attributes.
		
		Existing international standards for access of specific resource types will be used wherever applicable.

`Identity`	`Get Identities`	This interface connects with external identity management systems. OpenID or LDAP etc

		can be used to implement this. Defining this will be out of scope for this standard currently.
`Consent`	`Get consent from Provider`	This interface gets the consent from the Provider for accessing protected, private or confidential data. Since it involves interactions with Humans - it is not defined as part of this standard currently. SMS/Phone/Mail etc can be used. Not that in case of embedded `PII`, `Provider` has responsibility to get consent from `PII Principals`,

These interfaces are specified in detail in Part 2 of this standard.

There are no deployment restrictions imposed by the DX. Data exchange using the said reference architecture and the associated interface specifications, can be deployed in multiple different ways and DX does not favour or impose any specific deployment model.

4.4. Establishing Identities of Participants

The identities of the `Providers`, `App Developers` SHOULD be via X.509 certificate chains. The identity of `Consumers` MAY be via X.509 Certificates or ID tokens (OpenID Connect, SAML 2.0 or industry standards). The provisioning and management of these certificates or Tokens will be outside the scope of this standard.

4.5. Data Exchange Services

The two main services provided by the Data Exchange are the catalogue service, that allows management and search of meta-data about resources, and the Authorisation service, that manages authorisation to access the resources. These are described in more detail next.

4.5.1. Catalogue Service

On a high level the catalogue service enables the following:

- **Discovery of data resources:** By providing various search mechanisms to discover the resources of interest.
  - For example, geo-based search, text search on meta-information, attribute search with a given value etc.
- **Ease of consumption of data:** By providing links to **data models** that describe various attributes of the given resource. This facilitates data interoperability and easy integration into various applications.

 - Data-models may contain description of data types, units, value constraints, text descriptions, semantic context etc. for the data associated with a given resource. Further, the data-models may contain other meta-information about the resource which may not directly pertain to the data from the resource. For example, the location of a fixed sensor, device models etc.
- **Semantic modelling of meta-information:** By using linked data to provide semantic contexts for the attributes describing meta-information. This leads to improved machine readability, interpretability, operational interoperability and enables vocabulary reuse from other data-model stores and taxonomies.
- **Ease of access to data from a resource:** By providing formal descriptions of how to access the data from a resource.
    - For example, API objects to describe REST based resources etc.

At the core, DX catalogue is a store of *meta-information* associated with the data assets/resources available with the data exchange. A meta-information object may be related to another meta-information object by providing explicit references to one another. Further, using concepts of linked-data[1], semantic grounding is provided for the attributes contained in the meta-information objects. The catalogue service layer, built on top of the meta-information store, provides powerful search capabilities to discover resources of interest and their associated meta-information (e.g., data models, api objects etc.). Additionally, the catalogue provides services to build and maintain the meta-information store in a consistent and collaborative fashion.

Details of catalogue information model and various catalogue objects are provided in Section 6.

4.5.1.1. Catalogue Interface

Data Exchange catalogue exposes services via a set of APIs built on top of the meta-information store. The APIs can be broadly categorised into two sets:

- **Search** APIs to discover resource items of interest. The catalogue supports the following search methods:
    - Geo-spatial search: Discover items in a given geo-spatial bound (applicable to items containing geo-spatial attributes)
    - Attribute search: Discover items with a given value for a given attribute
    - Text search: Discover items that contain matching words in a set of textual attributes (e.g., text descriptions etc.)
- **Management** APIs to create, update and delete items.

The details of catalogue APIs are provided in part 2 of the data exchange standard.

4.5.2. Authorisation Service

4.5.2.1. Goals and Non-Goals

The main goal of data exchange framework (DX) is to enable seamless sharing of resources while respecting ownership, privacy and compliance requirements. DX achieves this by

[1] https://www.w3.org/DesignIssues/LinkedData

defining a set of open standards for authorization, data classification, and policy authoring, and providing sample implementations according to these standards. The standards enable data providers and application developers to target a consistent set of APIs for authoring policies and accessing data across smart city platforms. When DX is used for sharing sensitive or `PII`, the standards ensure that `PII principal`s retain control over data shared on the platform in accordance with the strongest privacy regulations.

The authorization service in DX is designed to reduce barriers for adoption. In particular, resource providers should be able to start sharing resources with authorized entities with minimal effort. Towards this end, DX will support mechanisms for plugging existing non-DX complaint resource and authorization servers into the DX ecosystem with simple extensions. The authorization service also supports a simple-to-understand data classification framework and policy authoring tools to help `provider`s migrate to the DX framework.

The following aspects of resource sharing are out of scope of DX authorization service.

- Data collection mechanisms: DX does not mandate how data is collected from edge devices and transferred to the resource server.
- Enforcement of policy mechanisms: DX does not currently provide mechanisms for enforcing policies such as data retention. Data providers are expected to enforce policies through other means such as legal agreements. However, the DX framework shall support technologies such as trusted execution environments for policy enforcement at the time of data use.
- Compliance requirements: DX does not mandate that policies meet specific compliance requirements. It is the responsibility of the `Provider` to ensure that the policy they define meets the required compliance requirements and laws. For example, DX does not mandate that a data provider only share information for which consent for sharing has been obtained.
- Information leakage: DX is not responsible for any leakage of information that may occur through sharing. `Provider`s are expected to ensure that data is appropriately sanitized e.g. via anonymization.
- It is outside the scope of DX to prescribe, what the consumer ought to do with the received data, post the retention period set by the provider. It is expected that the consumer disposes off the data in accordance with the policy requirements set by the provider, any legal agreement entered into with the provider or any regulatory framework governing the data.[DR(1]

4.5.2.2. Functionalities

The authorization service in DX **should** support the following functionalities.

- **Registration**. A resource provider should be able to register itself with the DX authorization service using X.509 certificates.

- **Resource Access Policy Management**. A registered resource provider should be able to register resources to be shared in the DX catalog service, and enable the DX authorization server to authorize access to resources on behalf of the provider. Resources **may** be associated with scopes and policies that define the set of resource attributes a consumer can access.
- **Resource Access Authorization**: A consumer application uses the information in the catalog to request access from a DX compliant resource server. The resource server in conjunction with the DX authorization server determines if the consumer should be granted data access. Once authorization has been obtained, subsequent requests for data access may be serviced entirely by the resource server.

4.5.2.3. Actors in the Authorization flow

The main actors of the architecture are:

- `Provider`: Data providers setup access control policies for resources they are serving via the policy interface exposed by the DX authorization server.
- `Consumer Application`: The consumer application requests access to data from the resource server on behalf of the consumer. Consumers are expected to obtain X.509 certificates rooted at a DX complaint certification authority.
- `Resource server`: The resource server hosts resources (data and services). It grants access to resources after validating tokens issued by the DX authorization server or the provider's own authorization server. If the resource server is IUDX/UMA 2.0 compliant and the request does not include a valid access token, the resource server initiates a IUDX/UMA 2.0 complaint authorization protocol involving the DX authorization server and the consumer application.

- `DX Adapter`: The DX Adapter is an optional entity that sits in front of a non-DX compliant resource server. It handles resource access requests from the consumer application on behalf of the resource server. If the resource server is not DX complaint, the Adapter initiates a IUDX/UMA 2.0 compliant authorization protocol involving the DX authorization server and the consumer application.

- `DX Authorization Server (DX AS)`: The DX authorization server is a IUDX/UMA 2.0 complaint auth server that can be configured to control access to resources on behalf of the `Provider`. It exposes policy, permissions, grant and token introspection endpoints.

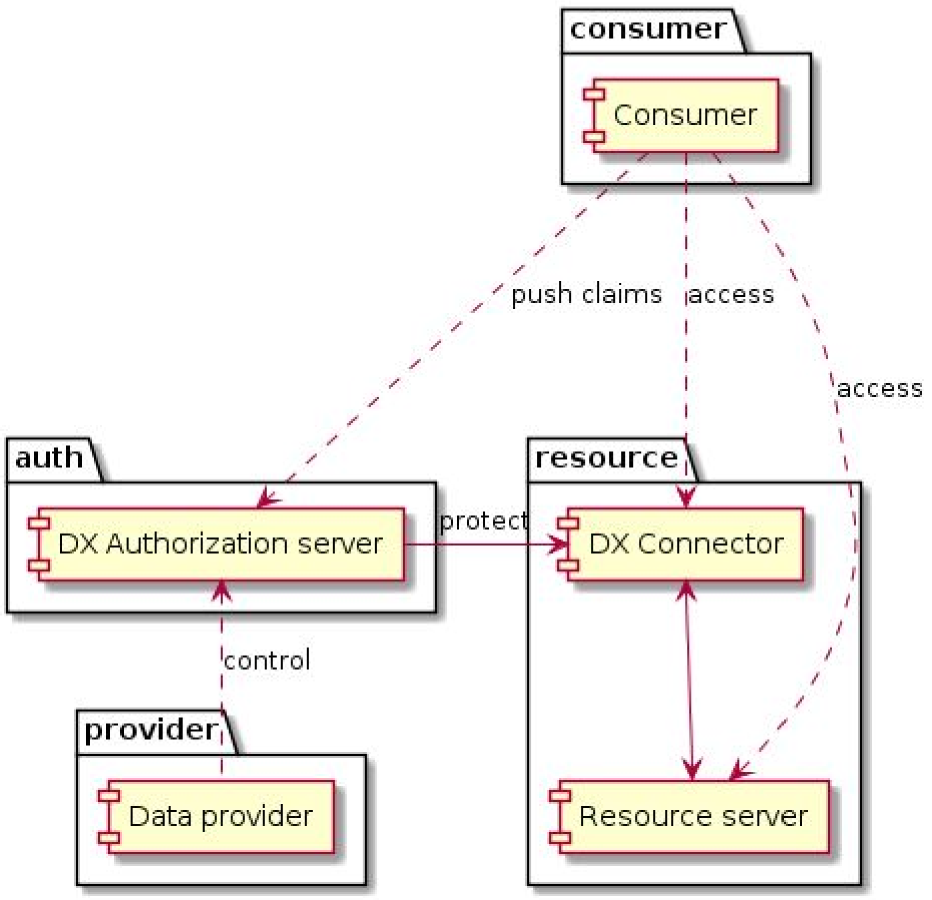

Figure 4-3: Authorization Flow

5. SECURITY AND PRIVACY

5.1. Security Considerations

- All interfaces **shall** use TLS for secure communications.
- A trusted CA, which is traceable to a root CA **shall** be used to provide certificates.
- Security Techniques specified in [i.13][i.14][i.15][i.16][i.17][i.18][i.19][i.20][i.21][i.22][i.23][i.24] **should** be followed.

5.2. Privacy Considerations

- All `resource-items` **shall** be tagged as either "public", "protected", "private", or "confidential".
- The protocols used to access resources **shall** always use secure protocols based on TLS.

- The authorization service **shall not** be able to know or tamper the contents of the consent artefact. Consent artefact should be encrypted by the Provider with the Consumers public key; and also signed by the Provider's private key.
- The consent artefact **shall** mention Consumer id, resource id to be accessed, permissions, start time, and end time of the artefact.

5.2.1. Out of Scope

The following aspects of data sharing are not in scope of DX.

- Data collection. DX does not mandate how data is collected from edge devices and transferred to the resource server.
- Enforcement of policies through technical means. DX does not currently provide mechanisms for enforcing policies such as data retention. Data providers are expected to enforce policies through other means such as legal agreements. However, the DX framework permits the use of technologies such as trusted execution environments for policy enforcement at the time of data use.
- DX does not mandate that policies meet specific compliance requirements. It is the responsibility of the data provider to ensure that the policy they define meets the required compliance requirements and laws. For example, DX does not mandate that a data provider only share information for which consent for sharing has been obtained.
- DX is not responsible for any leakage of information that may occur through data sharing. Data providers are expected to ensure that data is appropriately sanitized e.g. via anonymization.
- DX is not responsible for the correctness of the data sharing policy rules of a provider. The providers must ensure that their data policy rules match their organization's policies; and go through a review process, be vetted by concerned parties, and tested before applying.

5.3. Authentication and consent

- All Providers **shall** use certificates from DX-CA or recognized certificate authorities to connect with DX.
- The certificates **shall** be validated by authorization service and be checked against certificate revocation lists or OCSP.
- All Providers **shall** be able to manage the access control list of their resources.
- Catalog security: (i) Only people with proper authorization **shall** be able to create, update and delete entries in the catalog. (ii) The entries in the catalog are linked to the DN of a user's certificate. Thus, only the original owner **shall** be able to modify their own entry.

- Consent history: Consent history data is to be considered private and is only available to the Provider of the resource. Consent history **shall** be available to DX for audit purposes.

5.4. Authorization Policies

DX **shall** enable `Providers` to associate a policy with every `resource-item` in the catalog. The policy **should** govern who has access to the resource described by the item and how entities with access **should** handle storage of resource's data. Associating policy with attributes within a resource is currently not supported.This is because the resource is accessed at object granularity in DX. In order to associate policies at a finer granularity, providers **may** define views over underlying raw data, and associate a policy with every view.

A policy P is a pair of the form (W, H), where

1) W defines the set of consumers that are authorized for resource access. W may be defined in a policy language such as XACML or Aperture. A consumer is any entity such as an individual, organization, an organizational role, or a trusted execution environment that has been issued a certificate or token by a trusted CA.
2) H is a list of attribute-value pairs which defines requirements on consumers who have access to data.
3)

Table 5-1: Authorization policy attributes and values.

Attribute key	Description	Possible values
Authorization protocol and policy	Specifies protocol that a consumer must use to request access from a resource server	None, OAuth/UMA, OAuth/UMA + XACML Policy, Token/IUDX + Aperture policy language
Data locality	Specifies requirements on where a consumer is allowed to store data after getting access	Country, State, Organization (Service-based access), None
Data retention	Specifies requirements on how long consumers may retain data	Fixed period, Up to a certain event or date, None
Data storage	Specifies in what form a consumer may store data	Encrypted (using adequately protected keys), Encrypted (using keys owned by data owner), Any
Data usage	Specifies requirements on the purpose for which data is used.	Privacy preserving computation, Anonymization, Computation certified by a third party, Any
Data audit	Specifies audit requirements on data that the consumer must	Audit accesses along with time and duration of access, None

	satisfy	

This list is by no means complete. For example, the policy framework does not capture requirements around ownership of data. Further additions to the list of attributes or attribute values may be made over time while maintaining backward compatibility.

5.4.1. Policy labels

In order to simplify the process of authoring policies, we define a set of policy labels that capture commonly used policy specifications. The policy labels are only normative and may evolve over time.

Table 5-2: Some standard policy Labels

Labels → Meaning | v	**Public**	**Protected**	**Private**	**Confidential**
Nature of data	Contains no personal information	Contains anonymized information	May contain personally identifiable information	May contain personally identifiable information and/or other data that is confidential within the organization
Authorization protocol and policy	None	Requires authorization using IUDX/UMA, no custom auth policy	Requires authorization using IUDX/UMA, custom auth policy specified in a policy language	Requires authorization using IUDX/UMA, custom auth policy specified in a policy language
Consent	None	Provider	Requires consent of owners	Requires consent of owners
Data locality	None	None	Configurable or as per regulatory framework	Only service based access

Data retention	None	None	Configurable or as per regulatory framework	NA
Data storage	Any	Any	Encrypted	NA
Data usage	Any	License	Licensed with legal framework	Licensed with legal framework
Data audit	None	Random audit	Needs audit	Needs audit
Data Monetization	Not to be monetized	Provider's decision	Provider's decision	NA

5.4.2. Identity

Identity of producers/consumers of data in DX is through:

1. Certificates
2. Tokens

DX **shall** honour certificates from any licensed CA in India (certified by the CCA). Also, DX **may** issue certificates to users based on their email ids. DX **shall** host a certificate authority (CA) which will grant certificates to:

1. Resource servers
2. Individuals / App developers
3. Organizations
4. Data officers of an organization
5. Employees of an organization

Individuals, app developers, or employees of an organization who wish to access protected private, or confidential data **must** require a valid certificate.

Individuals and App developers **should** send a certificate signing request (CSR) as an attachment in an email to the `DX Administrator` with subject *"Certificate request"*. The DX CA will validate the CSR and will respond back with a certificate.

Organizations **should** send a certificate signing request (CSR) as an attachment in an email from their organization domain to `DX administrator`. The DX CA will validate the CSR and will respond back with a certificate. The certificate provided to organization can only be used to grant certificates to employees of the organization. Thus, the domain name of the organization must match with the e-mail of the employee for whom the certificate is granted.

For organizations to be able to request certificates from DX CA, they **shall** register themselves with DX CA (this **may** be through an online form). All registered organization's domain names **shall** be added to a white-list; and DX will only honour certificate requests from organizations in the white-list.

Organizations while generating a certificate **may** add more details about the employee such as: organization, organization unit, first name, last name, role of the employee, state, city, etc.

Employees of an organization may send a certificate signing request (CSR) as an attachment in an email to DX Administrator. The organization that will act as a sub-CA or a registration-authority will validate the CSR and will respond back with a certificate. The scripts to grant certificates **may** be provided by the DX to organizations.

DX CA **shall** issue 5 classes of certificates:

- Class 1: Which **shall** be issued to resource servers for validating tokens.
- Class 2: Which **shall** be issued to individuals or employees of an organization (which are registered and whitelisted with the DX). This certificate is expected to be used to access protected data.
- Class 3: Which can be issued to employees and data-officers of an organization (which are registered and whitelisted with the DX). Such employees **shall** have access rights to upload and manage `resource-items` in the catalogue of the DX. This certificate **shall** be used to create/manage catalog items.
- Class 4: Which is to be issued to trusted employees of an organization (which are registered and whitelisted with the DX). This certificate is expected to be used to access protected and private data.

- Class 5: Which is to be issued to trusted employees of an organization (which are registered and whitelisted with the DX). This certificate is expected to be used to access protected, private, and confidential data.

5.5. Audit Considerations

All Interfaces **should** make statistics available and log all events to allow audit of interactions.

5.6. Reliability Considerations

The systems must be designed to be highly available, scalable using a distributed architecture for vertical and horizontal scale, and high on performance. The APIs must have high uptime and a public API Status Page must be provided by the Authorization Service Provider, Catalogue Service Provider and the Resource Server that reports the same for each of the endpoints (along with other open data like average response time, latency, etc). Furthermore, these must also implement the Heartbeat API for reporting their system uptime in real-time.

This section explains how the various failure scenarios that must be handled [TBD]:

- Failure to notify `Provider`

  In this scenario, when the `AS` or `RS` is not able to notify the `Provider` on the status of the consent flow or data flow, a mechanism has to be put in place to notify the `Provider` at a later stage. This can be achieved by reinitiating the notification message to the `Provider` or by providing the `Provider` an option to check the status through an application, or by providing a list of all consent flows and data flows (with status) in the application.

- Response from `AS` does not reach the `Consumer`.

  In this scenario, when the response sent by `AS` does not reach the Consumer,the latter should have a mechanism provided by `AS` to initiate a request to know the status of the consent flows.

- Response from `Catalogue service` does not reach `Consumer`

  In this scenario, when the response sent by `catalogue service` does not reach the `Consumer`, the `Consumer` should have a mechanism provided by `catalogue service` to initiate a request to know the status of catalogue services.

- `RS` is not available to `Consumer`:

  In this scenario, when RS is not available to Consumer, Consumer may have a mechanism to re-initiate the request to RS.

6. IUDX catalogue

6.1. Catalogue Information Model

Conceptually, the catalogue is a collection of items with each item containing a set of meta-information ***attributes*** along with their ***values***. Each item belongs to an itemType which serves to categorise the type of meta-information contained in it. Each itemType is associated with a ‘schema’ which defines a mandatory set of attributes an item must contain. Other than the mandatory set of attributes, an item may contain additional custom attributes to augment its information content. IUDX catalogue supports the following itemTypes (see Table 6-1): ‘resourceItem’, ‘resourceServer’, ‘provider’, ‘resourceServerGroup’ and ‘catalogueItem’.

Each attribute in an item belongs to one of the attribute types chosen from a pre-defined set referred to as IUDX core attribute types. A core attribute type provides semantic context for an attribute and also defines the syntactic structure of that attribute. The syntactic structure is designed to be extensible such that additional information fields may be included apart from the minimal set of fields defined for a given core attribute type. IUDX catalogue supports the following core attribute types: ‘Property’, ‘Relationship’, ‘GeoProperty’, ‘QuantitativeProperty’ and ‘TimeProperty’.

The catalogue items are *linked data* objects. The mandatory attributes, and preferably the custom attributes as well, of an item are mapped to discoverable universal resource identifiers. That is, the attributes are provided with a *context*. The context for a given attribute may contain information on how the attribute should be interpreted. An attribute may contain linkages, using linked data primitives, to other attributes from other vocabularies/taxonomies thereby leading to further enhancement in its interpretability by machines (and humans). Further, different attributes from various items can point to the same resource identifier which provides a simple mechanism to harmonise semantically similar yet syntactically different attributes contained in different items.

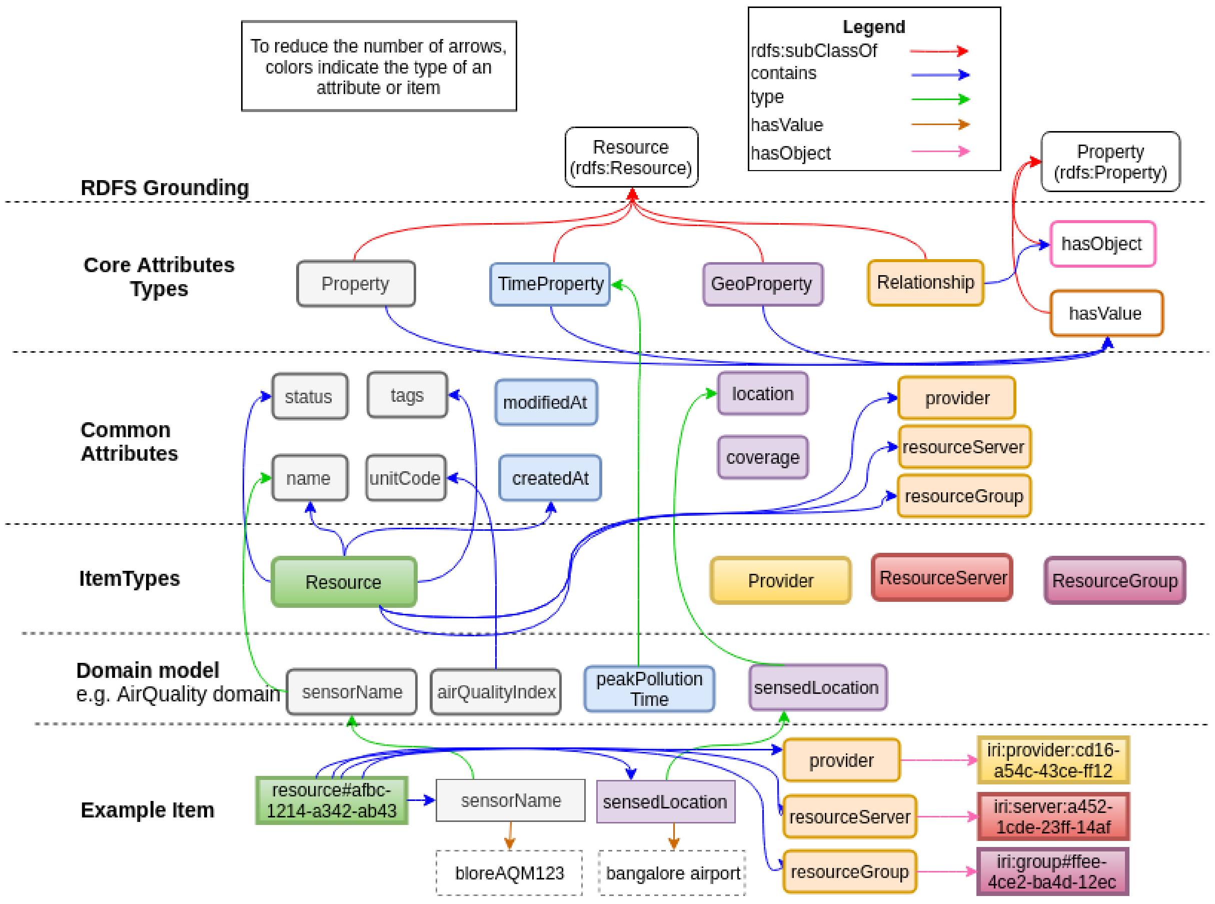

Figure 6-1 : IUDX Catalogue information model

Figure 6-1 summarizes the catalogue information model. The base layer consists of core attribute types. An attribute will, in general, have a ‘type’ and ‘value’ and may have other meta-properties associated with it to additionally describe the attribute. ‘type’ identifies the core attribute type and must have one of the following values: ‘Property’, ‘Relationship’, ‘GeoProperty’, ‘QuantitativeProperty’ and ‘TimeProperty’. ‘value’ contains the values assigned to the attribute and it may range from simple objects, like strings or numbers, to complex objects, like, ‘GeoJSON’ objects etc. IUDX core attribute types are described in Section 6.2.

Common attributes, which necessarily extend one of the core attribute types, serve to give specific meaning and interpretation to an attribute value. For example, ‘location’, which is of type ‘GeoProperty’, is used to specifically represent a geo-spatial point with just one set of latitude and longitude coordinates. Similarly, ‘createdAt’ is of type ‘TimeProperty’ which is used to represent the time (in a standard format) when a given catalog item is created. Common attributes have been defined to capture commonly used concepts across IUDX catalogue items. Further, all the mandatory attributes contained in various catalogue items necessarily come from the common attribute set.

An itemType is used to describe and give a semantic interpretation to a collective set of attributes contained in an item. For example, an item of type 'resourceItem' is used to describe meta-information associated with an IUDX Resource, like, how to access data from this resource, links to the data model used to describe data from this resource, discovery tags associated with this resource, information about the Provider of this data (who can authorize access) etc. Similarly, a 'provider' item captures information about Provider entity. As mentioned before, each itemType specifies a set of mandatory attributes to be included in any item of this itemType and these attributes necessarily belong to the common attribute set.

A data model describes meta-information attributes and their syntactic structure for a given application domain. The catalogue framework allows, using the concepts of linked data, reuse of attributes from other domain specific vocabularies. Note that defining a specific data model is out of scope for the catalogue specifications. However, for consistency purpose, attributes defined in data models should follow the same general set of rules, specified later, as are followed for IUDX meta-information attributes. The intent here is to specify a robust framework that allows specification, reuse and building consensus for domain specific data models.

6.1.1. Role of JSON-LD and JSON-schema

IUDX catalogue is based on JSON-LD[2] and JSON-schema[3]. JSON-LD framework is used to provide linked data encodings to the attributes in the catalogue items. The context mappings link attributes to vocabulary (or ontology) of interest which provides additional meaning and context to it. IUDX catalogue uses JSON schema framework for representational purposes. In particular, JSON schema is used for describing and validating the structure of catalogue items.

All the catalogue items are JSON-LD documents and necessarily need to include "@context" field, which contains JSON-LD context, that maps attributes to IRIs (Internationalized Resource Identifiers as described in [RFC3987][4]) providing unambiguous identification of these attributes. IUDX will necessarily provide context for IUDX core attribute types and IUDX common attributes. For additional attributes, it is recommended that the provider of these items should use context from IUDX vocabulary and/or from existing vocabularies, e.g., schema.org, GeoJSON-LD, etc.

JSON schemas are used to specify the structure of catalogue items. Each itemType has an associated JSON schema which is used to define the syntactic structure of an item belonging to this itemType. The schemas for all itemTypes is collectively referred to as base schemas.

JSON-schemas are also used to specify the domain specific data models. Data models describe attributes that contain meta-information about a data resource.

[2] https://json-ld.org/
[3] https://json-schema.org/
[4] https://www.ietf.org/rfc/rfc3987.txt

In addition, the JSON schema ‘definitions’ and/or ‘properties’ which are dereferenceable JSON pointers, and hence valid IRI links, can also be used to provide IRI references to be used in “@id” or “@type” fields from within the JSON-LD objects.

The base schemas and data models are not stored as catalogue items. However, a repository of base schemas and data models will be available to IUDX catalogue implementations.

6.2. IUDX Core Attribute Types

IUDX core attribute types represent a minimal set of types/classes to which various meta-information attributes belong to. Each core attribute has a well defined structure which is extensible to allow for additional information fields to be included for a given attribute. Since every attribute belongs to one of the core types, it imposes a partially known structure on various attributes, especially the ones that are not a part of IUDX common attribute set. Also, such an explicit categorisation allows for targeted operations on relevant core attribute types, e.g., geo-spatial search, time-based searches etc.

IUDX Catalogue information model supports the following core attribute types: **Property, Relationship, GeoProperty, TimeProperty, QuantitativeProperty**.

6.2.1. Property

An attribute of type ‘Property’ is a JSON object whose key is the attribute name, which is mapped to an IRI using “@context” field of the item containing the attribute, and whose value is a JSON-LD object that includes the following keys:

- “type”
  - Type “string” set to a fixed value “Property”
- “value”
  - Type “string”, “number”, “array” OR “object”[5]
- Any other attribute of core attribute types
- **Mandatory fields**: “type”, “value”

‘Property’ attribute is most general of the core-attribute types. In fact, as can be seen later, GeoProperty, TimeProperty and QuantitativeProperty are specializations of ‘Property’ that impose some additional structure on “value” field.

6.2.2. Relationship

An attribute of type ‘Relationship’ is a JSON object whose key is the attribute name, which is mapped to an IRI using “@context” field of the item containing the attribute, and whose value is a JSON-LD object that includes the following keys:

- “type”
  - Type “string” set to a fixed value “Relationship”

[5] In case the values takes by “value” is a JSON object, then it is assumed to be a JSON-LD node object and it is assumed that its context is either explicitly provided or the keys used are already mapped via the catalogue item context.

- “value”
  - Type “string” OR “array of strings”
  - Format: URI
- Any other attribute belonging to core attribute types
- **Mandatory fields**: “type”, “value”

‘Relationship’ attribute is useful to establish relationships amongst different catalogue objects. It can point to other items in the catalogue as well as external objects to which the items refer to, e.g., data models, base schemas, API objects, etc.

6.2.3. GeoProperty

An attribute of type ‘GeoProperty’ is a JSON object whose key is the attribute name, which is mapped to an IRI using “@context” field of the item containing the attribute, and whose value is a JSON-LD object that includes the following keys:

- “type”
  - Type “string” set to a fixed value “GeoProperty”
- “value”
  - Type: ‘object’ that includes one of the following
    - An ‘object’ with key “geometry” whose value is a GeoJSON object. The item should include the GeoJSON-LD context that maps the GeoJSON object keys to appropriate IRIs.
    - An ‘object’ with key “address” whose value is of type “string”
- Any other attribute belonging to core attribute types
- **Mandatory fields**: “type”, “value”

Attributes of type ‘GeoProperty’ are used to include geo-spatial information in catalogue items.

6.2.4. TimeProperty

An attribute of type ‘TimeProperty’ is a JSON object whose key is the attribute name, which is mapped to an IRI using “@context” field of the item containing the attribute, and whose value is a JSON-LD object that includes the following keys:

- “type”
  - Type “string” set to a fixed value “TimeProperty”
- “value”
  - Type “string”
  - Format: date-time in W3C format (default)
- Any other attribute belonging to core attribute types
- **Mandatory fields**: “type”, “value”

Attributes of type ‘TimeProperty’ are used to include time information in catalogue items.

6.2.5. QuantitativeProperty

An attribute of type ‘QuantitativeProperty’ is a JSON object whose key is the attribute name, which is mapped to an IRI using “@context” field of the item containing the attribute, and whose value is a JSON-LD object that includes the following keys:

- “type”
    - Type “string” set to a fixed value “QuantitativeProperty”
- “value”
    - Type “number” or “array of numbers”[6]
- “unitCode”
    - Type “string”
        - Denote the units of measurement (https://schema.org/unitCode)
- “unitText”
    - Type “string”
        - Additional textual description for the units of measurement. Very useful when unitCode is not available for the quantity of interest. (https://schema.org/unitText)
- “minValue”
    - Type “number”
        - Lower numeric limit of the described quantitative property (https://schema.org/minValue)
- “maxValue”
    - Type “number”
        - Upper numeric limit of the described quantitative property (https://schema.org/maxValue)
- Any other attribute belonging to core-attribute types
- **Mandatory fields**: “type”, “value”

Attributes of type ‘QuantitativeProperty’ are useful to represent observed or measured quantities.

In terms of property graph model the following (non-normative) mappings could be made:

- Each of the core attribute types can be perceived as sub-class of ‘rdfs:Resource’[7].
- The field “value” maps (via JSON-LD mappings) to ‘hasValue’ which is of type ‘rdfs:Property’[8]
- The field “object” maps (via JSON-LD mappings) to ‘hasObject’ which is of type ‘rdfs:Property’

The IUDX catalogue core attribute types are similar to the core meta-model in NGSI-LD[9], a recent data exchange standard by ETSI ISG CIM, which incorporates the linked data concepts in the data served by Resource Servers. In particular, ‘Property’ and ‘Relationship’ objects are similar between IUDX catalogue and NGSI-LD core meta-model. Further, the

[6] A numeric quantity represented as a string is acceptable. Any non-numeric characters in the string will not be accepted.
[7] https://www.w3.org/TR/rdf-schema/#ch_resource
[8] https://www.w3.org/TR/rdf-schema/#ch_property
[9] https://www.etsi.org/deliver/etsi_gs/CIM/001_099/009/01.01.01_60/gs_CIM009v010101p.pdf

core-attributes ‘GeoProperty’ and ‘TimeProperty’ are also in consonance to concepts defined in the NGSI-LD common meta model.

6.3. IUDX Common Attributes

IUDX common attributes define commonly used concepts in IUDX catalogue items. All the common attributes belong to one of the core attribute types and thus follow the structure of the parent attribute type. Informally, the common attribute definition adapts the core attribute type to a specific concept that is being modelled. For example, although ‘location’ and ‘coverageArea’ are both of type ‘GeoProperty’ these are modelling different geo-spatial scenarios. Whereas, ‘location’ represents a point by restricting the ‘geometry’ object to be of type ‘GeoJSON point’, ‘coverageArea’ represents a geo-spatial region by restricting the ‘geometry’ object to be of type ‘GeoJSON polygon’.

The syntactic structure for common attributes is defined using JSON schemas. As mentioned before, the attribute definitions from within the JSON schema document may be used as the corresponding IRI link for the linked data context.

Table 6-2 lists down all the common attributes defined within IUDX catalogue.

6.4. Base schemas, data models and API objects

6.4.1. Base schemas

As mentioned before, each catalogue item belongs to an itemType. Further, all itemTypes have associated schemas, collectively referred to as base schemas, which describe the syntactic structure of the items belonging to the respective itemTypes. The base schemas are represented using **JSON-schemas**. A base schema specifies the template of a given item which includes: specifying the set of common attributes used in an item, specifying a set of mandatory attributes, specifying constraints (if any) on additional attributes etc.

By specifying a set of mandatory attributes base schemas ensure a level of uniformity in the information contained in an item of a given itemType. For example, for every ‘resourceItem’, a mandatory ‘tags’ field is useful to search for resources of interest. At the same time the flexibility of including additional attributes allows for including specific information in an item that may be hard to generalise.

Representing base schemas as JSON-schemas brings into play the powerful validation framework that can be leveraged to validate items at the time of creation or updation of items into the catalogue store.

With regards to any additional attribute included in any item, it is recommended that the general guidelines be followed:

- It should belong to one of the core attribute types.
- Context for the new attribute, i.e., IRI mapping and “@type” definitions, should be added to the item context. Also, as mentioned before, if the ‘value’ of the new attribute is a JSON-object and it needs to be treated as an LD object, then its context should be provided.

The catalogue contains the following item types:

Table 6-1: Item types in a DX catalogue

itemType	Description
resourceItem	Contains meta-information associated with a data resource.
resourceServer	Contains information about a Resource server.
resourceServerGroup	Contains information about a resource server group[10].
provider	Contains information about the IUDX Provider entity
catalogueItem	Contains information about the "catalogue" instance. For example, instance id, end points, Provider etc.

Table 6-3 lists down the set of mandatory attributes for each of the above itemType[11]. Also see Section 6.4.4. for a discussion on the relationship between various catalogue itemTypes and other catalogue objects.

6.4.2. Data Models

The data model object contains description of domain specific attributes associated with a resource. These can include attributes that describe the data from resource, also referred to as *data-attributes*, as well as attributes that describe other meta-information related to the resource. As an example, let the data from a temperature sensor be available in JSON format and contain keys "temperature" and "time-stamp". The data model corresponding to this sensor must contain schemas for attributes "temperature" and "time-stamp" along with optional textual descriptions about these attributes. Further, the data model may also include attributes, like, location of sensor, make and model of sensor etc. which do not directly describing the data and yet contain important meta-information related to the resource.

JSON schema is used to represent data models in IUDX catalogue. The attributes contained in the data model, in a fashion similar to IUDX common attributes, must belong to one of the core attribute types. Also, wherever applicable, data model attributes should use attribute definitions from the IUDX common attribute set.

Data models improves understanding and interoperability of data from a given resource. Further since JSON schema framework is used to represent data models, these can also be used for validation purposes by data consuming applications.

In IUDX catalogue framework the data model, which is a JSON schema document, also plays a role in enabling linked data. The JSON-LD "@context" field containing the context for

[10] A resource group defines a grouping of resources within same Resource server and share same data model. See Section 6.4.4. for more details.
[11] Also refer to the IUDX Github repo (https://github.com/iudx/iudx-ld) for latest releases of base schemas and data models.

all the data model attributes is included in the data model. This will enable easy conversion of JSON data from resource server into a JSON-LD data. All that is required is to add an “@context” field in the JSON data from the resource containing a reference to the corresponding data model.
Note that by adding “@context” field the data model serves dual purpose: It can be referred to as a valid JSON-LD context document and all fields outside “@context” will be ignored. Similarly, it remains a valid JSON schema document and when used as a schema document, say for validation purposes, all non JSON schema fields will be ignored.

The idea of adding “@context” is further extended in the following way: The data models may include some ‘non JSON schema attributes’ in the schema and a JSON-LD context is provided for these attributes. The JSON-LD parser will expand these to their corresponding IRIs thereby providing the consuming applications with additional context about a given data attribute. This leads to improved human and machine interpretability of these attributes and eventually a better understanding of data itself.

The data models are not stored as catalogue items. However, a repository of base schemas and data models will be available to IUDX catalogue implementations.

6.4.2.1. Compact representation of attributes

As mentioned above, the data models include data attributes which belong to one of the core attribute types. It may not always be possible (and even preferable) to send data according to the template required by the associated core type. For these scenarios, a compact representation is supported where the data attribute and its ‘value’ are represented as a simple key-value pair (thus, excluding the explicit mention of fields ‘type’ and ‘value’). However, one can easily get the ‘type’ of this attribute using the JSON-LD context mappings and/or by examining the data model corresponding to this data resource.

6.4.3. Access Objects

The access object formally describes the methods of accessing data from (and interacting with) the resource. For example, an open-API object[12] can be used to describe API end-points, query parameters, request and response bodies for a REST API based access. Reference for such objects can be included in catalogue resource items to ease the process of accessing data from a given resources or a set of resources. Other such formal descriptions, e.g., async API objects[13] for MQTT/AMQP messaging based resources etc. can also be included in the resource items.

An interesting aspect is that in some access objects, e.g., API object, the request/response bodies may be described using JSON-schema references. For such access objects, it is recommended that the access object refer to the corresponding data model (which is a JSON-schema) to describe attributes in the request/response bodies.

The access objects are not stored as catalogue items.

[12] https://swagger.io/docs/specification/about/
[13] https://www.asyncapi.com/

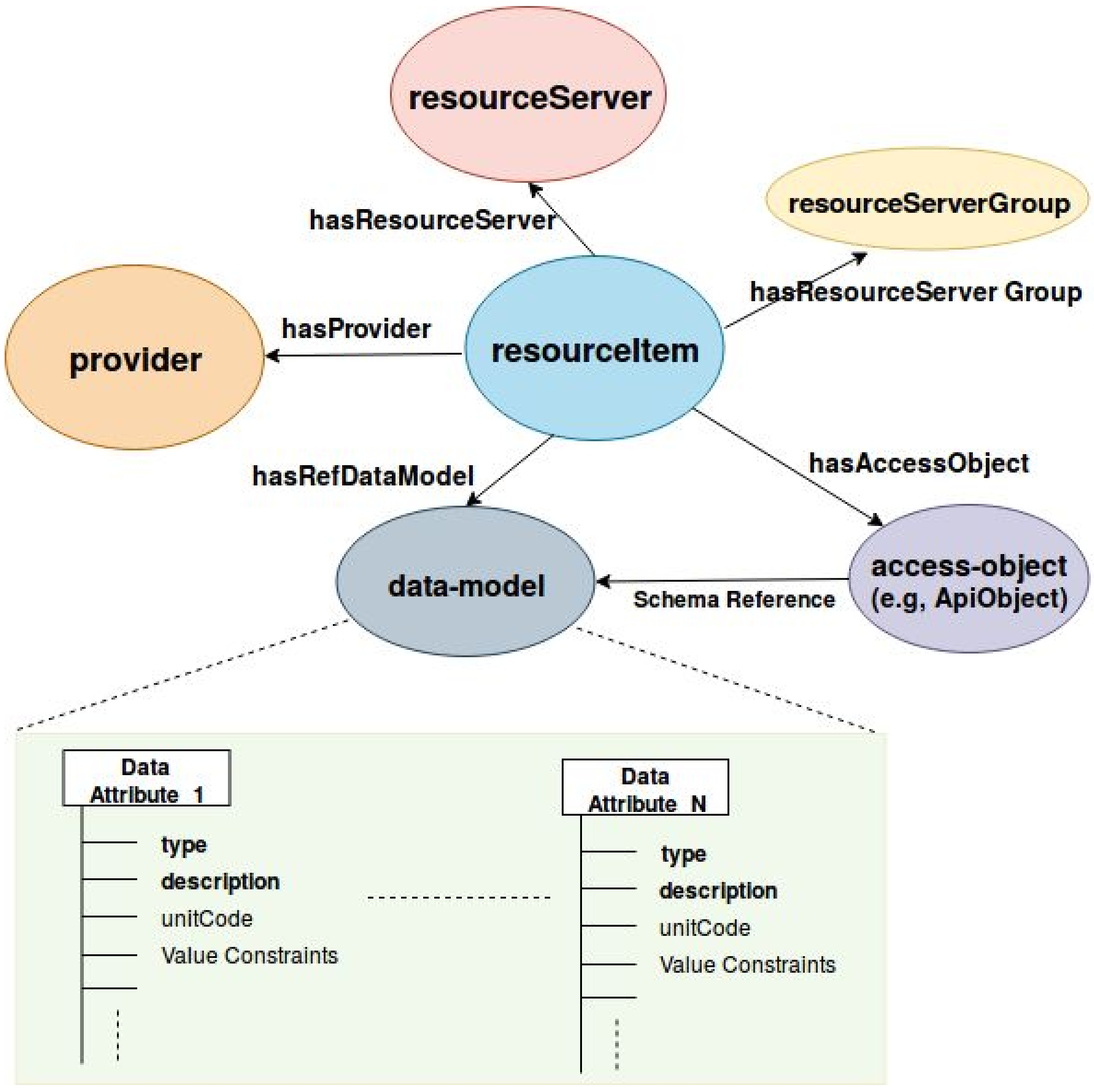

Figure 6-2: Relationship of "resourceItem" with other itemTypes

6.4.4. Relationship between various catalogue objects

Figure 6-2 summarizes the relationship between various IUDX catalogue objects.

An item of type 'resourceItem' is a key object around which catalogue services have been designed. It consolidates different types of meta-information associated with a resource. For example, links to the associated data model objects, links to associated API objects etc.

In addition to the attributes specified by the base schema, a resource item may contain additional attributes from an associated data model, e.g., location information of a fixed sensor or device model and manufacturer etc.

An item of type 'resource-item' **must** contain a reference to an item of type 'provider'. The provider object contains information about the provider for this resource, e.g., identity of the

Provider, description, contact information etc.. This information is needed for applications looking to get authorization to access this resource.

Similarly, an item of type ‘resourceItem’ **must** contain a reference to an item of type ‘resourceServer’ which contains information about the Resource server that is hosting the resource, e.g., identity, description, name, IP address and ports etc.

Additionally, a ‘resourceItem’ may contain a reference to an item of type ‘resourceServerGroup’ which identifies the resource group within a resource server to which the resource belongs. A resource group is a grouping of resources that have the same data model, same access objects (and hence the same request/response bodies) and belong to the same Resource server. The concept of resource groups allow operations (e.g., get data, status etc.) to be defined on multiple resources belonging to the same group. Currently, no notion of resource groups exist across the resource servers. However, once the data models for certain resource groups have been standardized it can be envisioned that a resource group can have resources from different resource servers also. The ‘resourceServerGroup’ item contains information that is applicable to all the resources within this group and hence this information need not be included in each resource item.

Table 6-2 below lists down the attributes defined in the IUDX common attribute set. These attributes are used to define base schemas. Table 6-3 lists down the common attributes used in IUDX base schemas.

Table 6-2: List of IUDX common attributes

Field Name	**Attributes**	**Value**
	type	Property
	description	id of a catalogue item
id	value-type	string
	type	Property
	description	id of the resource in the resource server
resourceId	value-type	string
	type	Property
	description	name of a catalogue item
name	value-type	string
createdAt		

	type	TimeProperty
	description	Time when a resource/item/attribute was created
	value-type	string
modifiedAt		
	type	TimeProperty
	description	Time when a resource/item/attribute was modified
	value-type	string
deprecatedAt		
	type	TimeProperty
	description	Time when a resource/item/attribute was deprecated
	value-type	string
tags		
	type	Property
	description	Array of keywords describing this item facilitating item discovery.
	value-type	array
resourceType		
	type	Property
	description	Type of the resource (see Table 6-4 below)
	value-type	string
uriLink		
	type	Property
	description	Attribute whose value is a URI
	value-type	string
itemStatus		
	type	Property
	description	Status of this catalogue item
	value-type	string
refBaseSchema		
	type	Relationship
	description	Link to the base schema for this itemType
	value-type	IRI
resourceServer		
	type	Relationship
	description	Link to a resourceServer item

	value-type	IRI
	type	Relationship
	description	Link or an array of links to resourceServerGroup items
resourceServerGroup	value-type	IRI
	type	Property
	description	Text description of this item.
itemDescription	value-type	string
	type	Property
	description	iudx item type
itemType	value-type	string
	type	Relationship
	description	Reference to the data model for this resource item.
refDataModel	value-type	IRI
	type	Relationship
	description	Link to the provider of this resource
provider	value-type	IRI
	type	Property
	description	Status of an item. Set to either 'active' or 'deprecated'
statusSchema	value-type	string
	type	GeoProperty
	description	Describes a geo-spatial location as a geoJSON point
location	value-type	object
	type	GeoProperty
	description	Describes a geo-spatial region as a geoJSON polygon
coverageRegion	value-type	object
organizationInfo		

	type	Property
	description	Information about a given organization (contact info, email, urls etc.)
	value-type	object
	type	Property
	description	Information regarding the authorization server that this item uses
authorizationServerInfo	value-type	object
	type	Property
	description	Device model information, it's make, brand, model, url, etc
deviceModelInfo	value-type	object
	type	Property
	description	Type of access mechanism. For example, 'openAPI', 'asyncAPI', 'custom'.
accessObjectType	value-type	string
	type	Property
	description	URL that points to more information about data access of this resource
accessObjectURL	value-type	string
	type	Relationship
	description	Link to an object (OpenAPI 3.0 api JSON object, or a json-schema) to describe access mechanism for this data resource.
accessObject	value-type	IRI
	type	Property
	description	Item specific API object variables. The variables and their corresponding value for this resource item are listed as a key-value pairs in value field of this property. The json-object in the value should be treated as a simple json object and not a json-ld object.
accessObjectVariables	value-type	object

accessInformation		
	type	Property
	description	List of access mechanisms available for data associated with this catalog item
	value-type	array
dataAttributeList		
	type	Property
	description	Array of fields from the data-model which appear in the data packet. These fields are not necessarily instantiated in the resourceItem
	value-type	object

Table 6-3: Mandatory attributes for IUDX base schemas

itemType	**core attributes**
providerItem	id, name, tags, refBaseSchema, itemDescription, itemType
resourceItem	id, tags, refBaseSchema, resourceServer, itemDescription, refDataModel, provider, resourceServerGroup, resourceId, itemType
resourceServer	id, name, tags, refBaseSchema, itemDescription, resourceServerHTTPAccessURL(uriLink), resourceServerOrg(organizationInfo), coverageRegion, itemType
resourceServerGroup	id, name, tags, refBaseSchema, resourceServer, itemDescription, refDataModel, provider, itemType

Table 6-4: List of supported resource types

#	**Nature of Resource**	**Type**
1	Data set as a file	`file`
2	Data set as a table of records	`table`
3	Data as a notification (e.g alert, event)	`message`

4	Data as a stream of messages (e.g. sensor readings)	`message Stream`
5	Media stream (temporally encoded like video or audio)	`media stream`

7. INTERACTION SCENARIOS

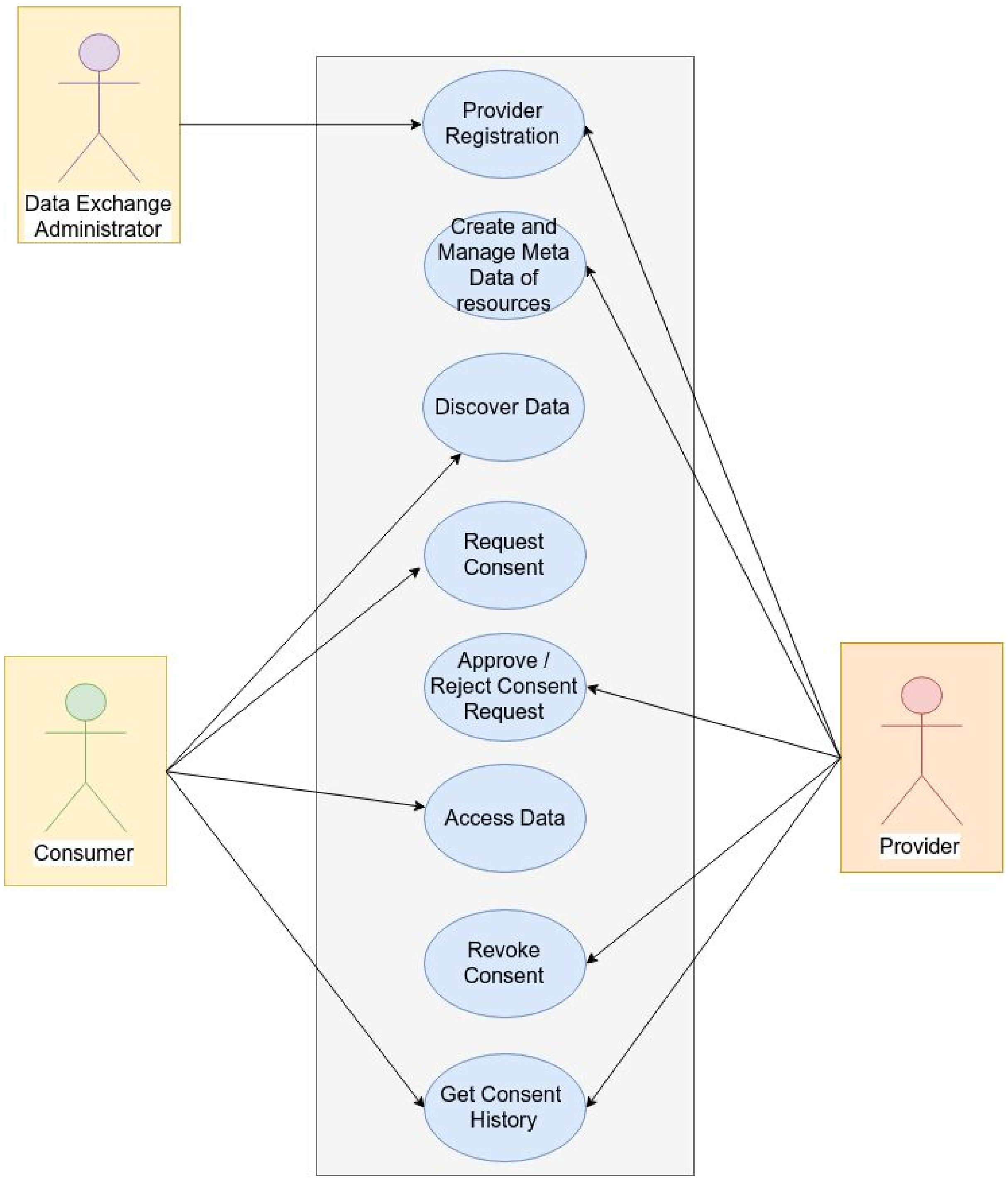

Figure 7-1: Basic Interaction Scenarios

The minimal set of interaction scenarios supported by the data exchange ecosystem is indicated in the Figure 7-1. Details for each follows.

7.1. Provider Registration

The provider needs to obtain a certificate from the DX CA to establish identity. An example workflow is given as follows

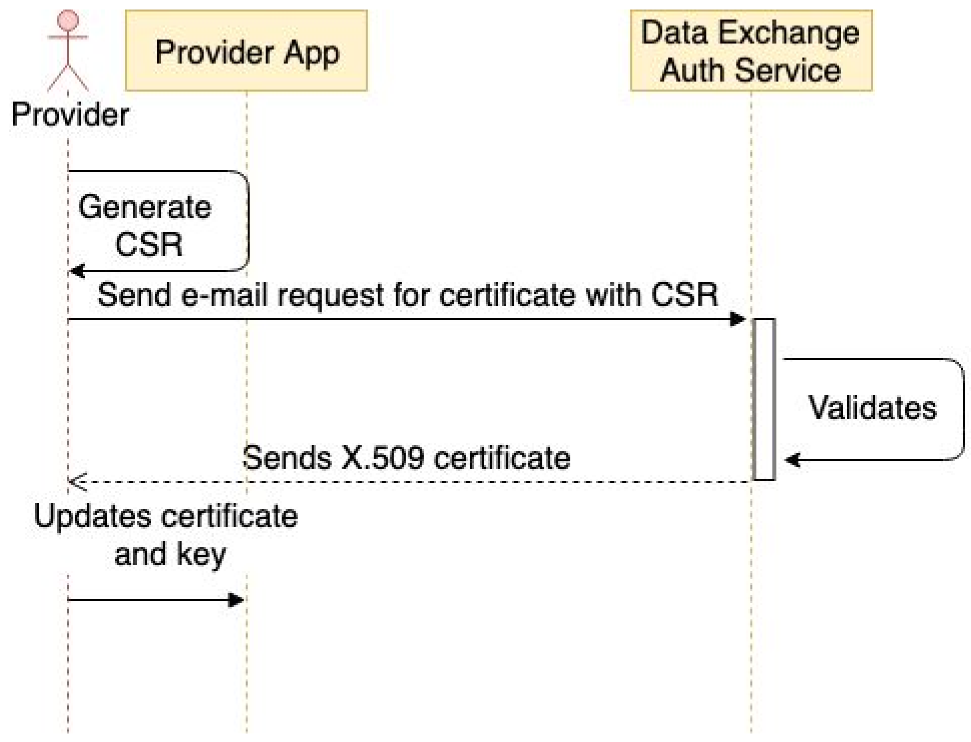
Provider
Provider App
Data Exchange Auth Service
Generate CSR
Send e-mail request for certificate with CSR
Validates
Sends X.509 certificate
Updates certificate and key

Figure 7-2: Provider registration flow

7.2. Create and manage metadata of resources

In this use case, a `Provider` is requesting the `Authorisation Service` to allow access to use the Catalogue. Once approved, the `Provider` can create or update an entry in the catalogue.

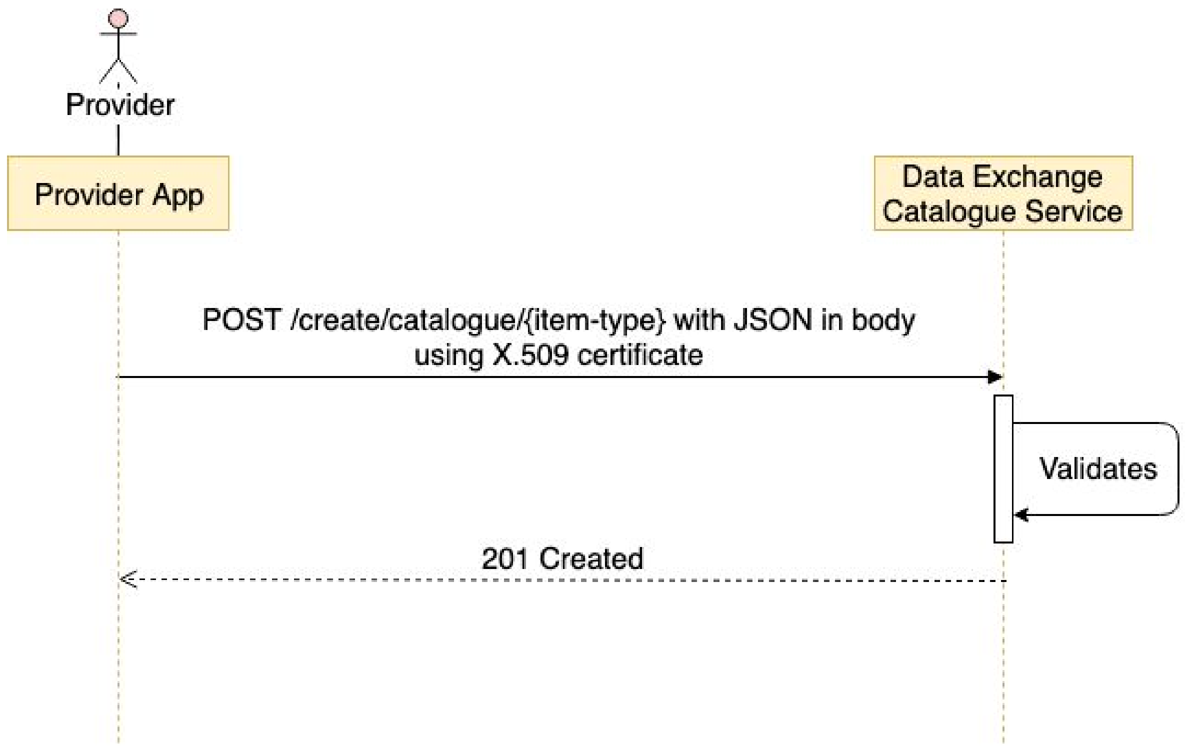
Provider
Provider App
Data Exchange Catalogue Service
POST /create/catalogue/{item-type} with JSON in body using X.509 certificate
Validates
201 Created

Figure 7–3: Resource Provider getting access to use Catalogue

Summary	The use case allows Providers to update the catalogue
Pre-condition	● A valid certificate provided by a trusted CA
Actor	Provider
Post-condition	An approval is provided to perform the requested operation

If a resource is not public, then the `Provider` can request an `Authorisation Service` to set policies for data access.

Summary	The use case allows Providers to set policies for their resources
Pre-condition	● A valid certificate provided by a trusted CA
Actor	Provider
Post-condition	A policy is set for the provider's resources.

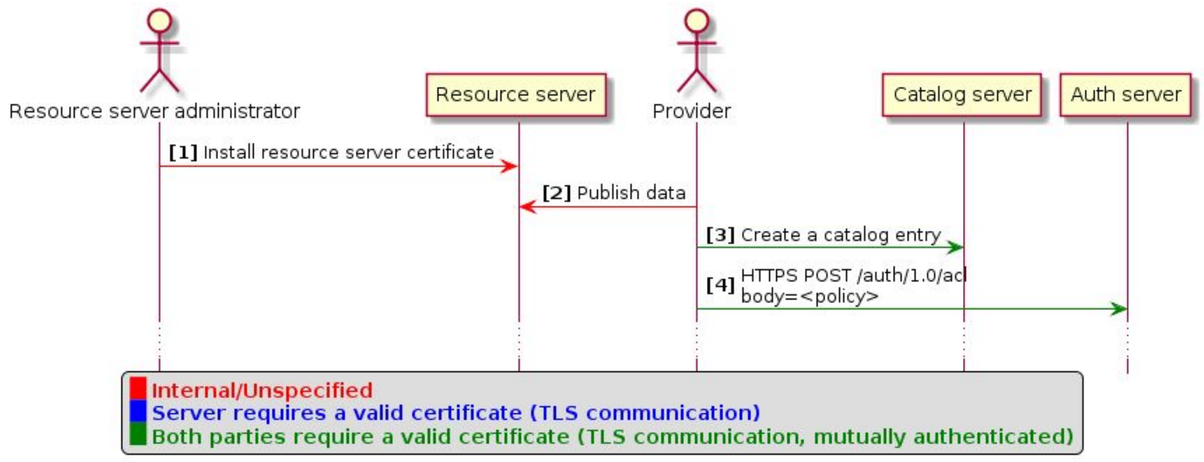
Resource server administrator
Resource server
Provider
Catalog server
Auth server
[1] Install resource server certificate
[2] Publish data
[3] Create a catalog entry
[4] HTTPS POST /auth/1.0/acl body=<policy>
Internal/Unspecified
Server requires a valid certificate (TLS communication)
Both parties require a valid certificate (TLS communication, mutually authenticated)

Figure 7–4: Setting up a permissions for a resource by the provider

7.3. Discover Data

In this use case, a `Consumer` is using the search APIs of the `Catalogue Service` to find interested entities. Once the entities are identified, a `Consumer` using `Consumer App` can request for consent and access their data.

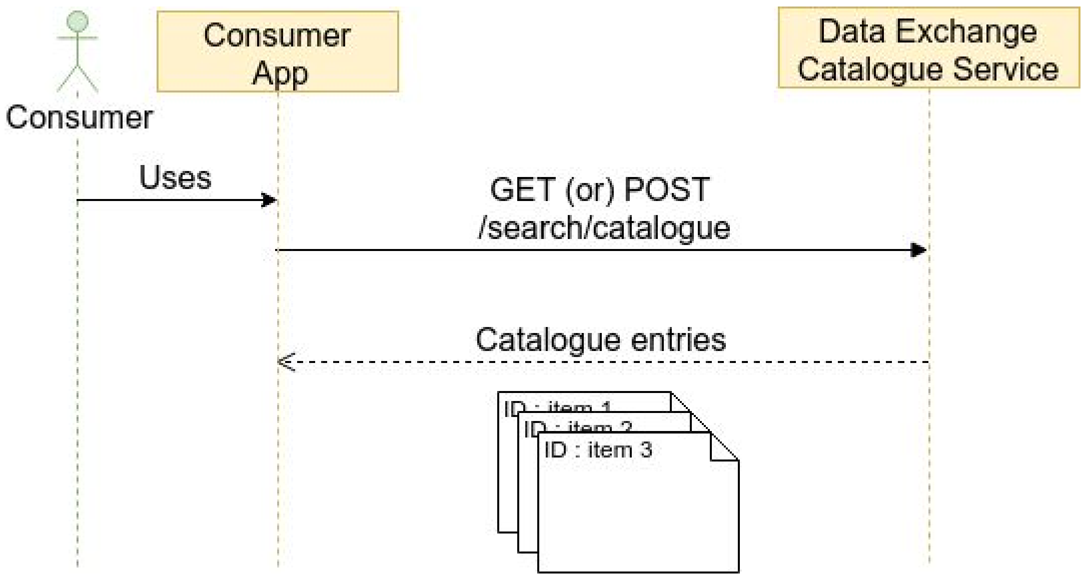

Figure 7–5: Consumer discovering data

Summary	This use case allows consumers to search the catalogue using

	customer app
Pre-condition	-
Actor	Consumer
Post-condition	A list of search hits are sent to the consumer app

7.4. Request Consent and Access Data

A consumer application can request access to data from an DX compliant resource server using any one of the supported APIs. If requested data does not require authorization i.e. does not have an authorization policy, or the request contains a valid access token, then the resource server serves the request after token validation.

If data requires authorization and the request does not contain a token, the resource server initiates authorization following a IUDX/UMA 2.0 compliant protocol. The protocol supports a subset of UMA 2.0 workflows.

1) Accept policies specified in a policy language.
2) Support identities based on X.509 certificates issued by DX CA (as described above)
3) Accept claims based on X.509 certificates issued by a set of trusted CAs
4) No support interactive claims gathering. All claims must be pushed.
5) Include a reference to the policy object in access tokens issued

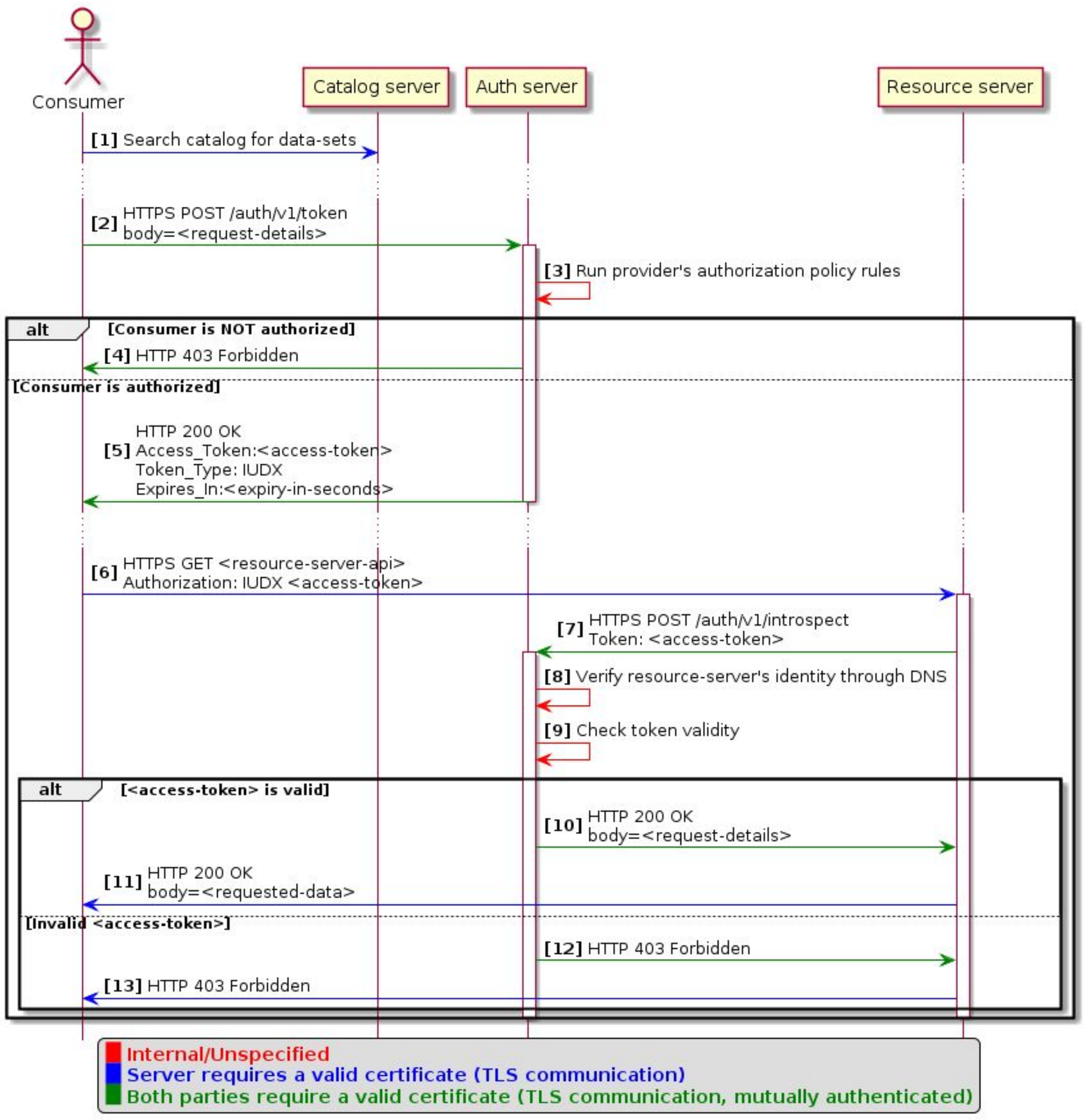
Consumer
Catalog server
Auth server
Resource server
[1] Search catalog for data-sets
[2] HTTPS POST /auth/v1/token body=<request-details>
[3] Run provider's authorization policy rules
alt [Consumer is NOT authorized]
[4] HTTP 403 Forbidden
[Consumer is authorized]
[5] HTTP 200 OK Access_Token:<access-token> Token_Type: IUDX Expires_In:<expiry-in-seconds>
[6] HTTPS GET <resource-server-api> Authorization: IUDX <access-token>
[7] HTTPS POST /auth/v1/introspect Token: <access-token>
[8] Verify resource-server's identity through DNS
[9] Check token validity
alt [<access-token> is valid]
[10] HTTP 200 OK body=<request-details>
[11] HTTP 200 OK body=<requested-data>
[Invalid <access-token>]
[12] HTTP 403 Forbidden
[13] HTTP 403 Forbidden
🟥 Internal/Unspecified
🟦 Server requires a valid certificate (TLS communication)
🟩 Both parties require a valid certificate (TLS communication, mutually authenticated)

Figure 7–6: Consumer requesting access to a resource

Summary	This use case allows the Consumer to access the requested data
Pre-condition	● A valid X.509 certificate issued by a DX compliant CA
Actor	Consumer Application
Post-condition	The requested data is provided to the Consumer

7.6. Revoke Consent

A provider **shall** be able to revoke access to a particular resource by calling the /revoke API.

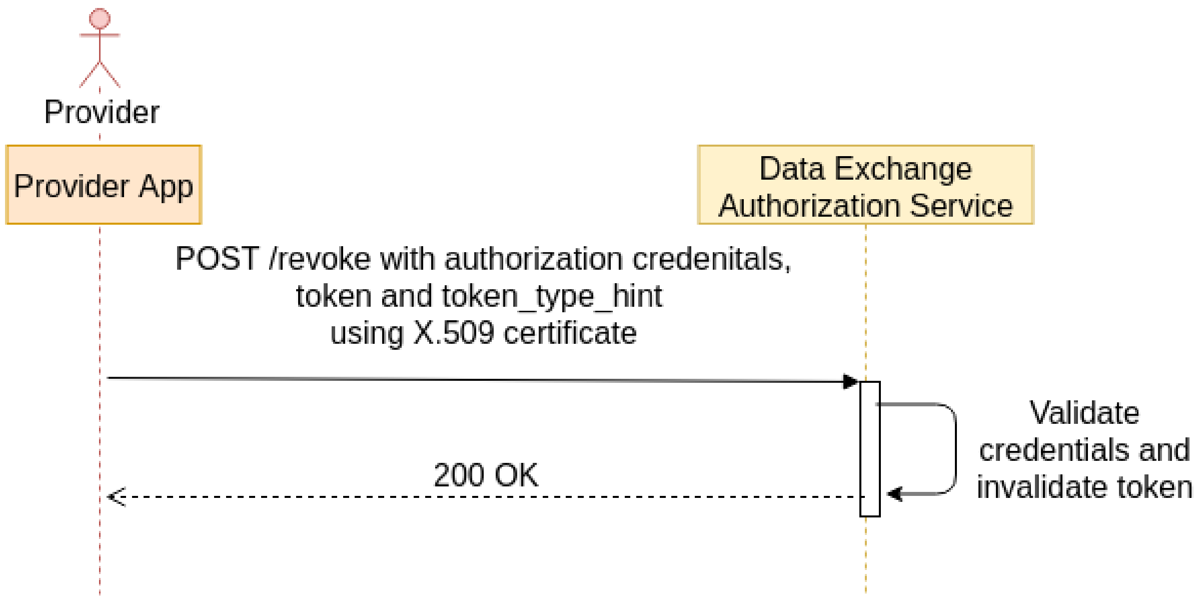
Provider
Provider App
Data Exchange Authorization Service
POST /revoke with authorization credenitals, token and token_type_hint using X.509 certificate
200 OK
Validate credentials and invalidate token

Figure 7–7: Revoking consent flow

Summary	This use case allows the Provider to revoke access to data
Pre-condition	● A valid X.509 certificate issued by a DX compliant CA
Actor	Provider Application
Post-condition	The consumer is no longer able to access the resource

8. Annex 1: Examples of catalogue objects

As described before, all catalogue items are JSON-LD objects and are instances of pre-defined JSON-schemas. The catalogue contains items of different itemTypes, each having an associated base-schema. A base-schema specifies a minimal set of mandatory meta-information attributes to be contained in the catalogue item. Further, a ‘resourceItem’ is associated with a data-model, which is a JSON schema document, and an API access object. In this annexe, we provide examples of various catalogue objects.

Let us take an example of an item with ‘itemType’ ‘resourceItem’ that corresponds to data observed from a physical sensor device measuring ‘CO2’ and ‘TEMPERATURE’.

First, we provide an example of data-model describing the domain specific attributes of the sensor device. One can create a new data model[14] from scratch or one can reuse some existing data models for devices in the same domain.

<catalogue-link>/airQuality/airQuality_dataModel.json

```
{
   "$schema": "http://json-schema.org/draft-07/schema#",
   "@context": [
       "<catalogue-link>/core_context.json",
       "<catalogue-link>/common_context.json",
       "<catalogue-link>/miscSchemaOrgDefs.json",
       "<catalogue-link>/miscWOTSecurityDefs.json",
       "<catalogue-link>/geometry-schema.json",
       "<catalogue-link>/airQuality/airQuality_context.json"
   ],
   "describes": "Environmental Sensor measuring CO2 and Temperature",
   "type": "object",
   "properties": {
       "CO2_MAX": {
           "$ref": "<catalogue-link>/core_defs.json#/definitions/QuantitativeProperty",
           "describes": "Maximum value of CO2 for the last 24 hours",
           "symbol": "ppm",
           "unitCode": "X59",
           "unitText": "part per million (ppm)"
       },
       "LASTUPDATEDATETIME": {
           "$ref": "<catalogue-link>/core_defs.json#/definitions/TimeProperty"
       },
       "NAME": {
           "$ref": "<catalogue-link>/core_defs.json#/definitions/Property",
           "describes": "Name of the device."
       },
       "TEMPERATURE_MAX": {
           "$ref": "<catalogue-link>/core_defs.json#/definitions/QuantitativeProperty",
           "describes": "Maximum value of Temperature for the last 24 hours",
           "unitCode": "CEL",
           "unitText": "degree Celsius",
           "minValue": -20,
```

[14] Existing data-models can serve as examples or as a starting point for creating newer data models. In future, tools may be provided to ease data-model development.

```
            "maxValue": 50
        },
        "location": {
            "$ref": "<catalogue-link>/common_defs.json#/definitions/location",
            "describes": "Describes the coordinates (in GeoJSON Point type) for the installation location
of the AQM device"
        }
    }
}
```

The context for attributes in the datamodel are derived from a json document mentioned in the “@context” field of the above, namely “<catalogue-link>/airQuality/airQuality_context.json".

<catalogue-link>/airQuality/airQuality_context.json

```
{
    "@context": {
        "airQuality_context": "<catalogue-link>/airQuality/airQuality_context.json#/",
        "CO2_MAX": {
            "@id": "airQuality_context:CO2_MAX",
            "@type": "QuantitativeProperty"
        },
        "TEMPERATURE_MAX": {
            "@id": "airQuality_context:TEMPERATURE_MAX",
            "@type": "QuantitativeProperty"
        }
},
    "CO2_MAX": {
        "describes": "Maximum value of CO2 for the last 24 hours"
    },
    "TEMPERATURE_MAX": {
        "describes": "Maximum value of Temperature for the last 24 hours"
    }
}
```

The above data model describes various attributes of the sensor resource, e.g., ‘NAME’ of the device, ‘location’ of the device etc. Further, for ‘TEMPERATURE_MAX’ and ‘CO2_MAX’, which are data attributes, additional information, e.g., ‘unitCode’, ‘unitText’, ‘minValue’, ‘maxValue’ etc., has been provided.

Note that the above additional information keywords, e.g., ‘unitCode’, ‘unitText’ etc., are not JSON schema keywords and will be ignored by JSON schema tools/validators. However, these keywords have been provided linked data grounding using “@context” field in the data model. If this data-model is passed through a JSON-LD parser, these keywords will expand to the IRIs provided via “@context” and will serve to provide additional context about these properties. A snippet of JSON-LD expanded object for ‘TEMPERATURE_MAX’ is shown below:

<TEMPERATURE_MAX>

```
{
.
.
  "https://<IRI for TEMPERATURE_MAX>": {
        "https://schema.org/maxValue": {
        "@type": "https://<IRI for core_defs.json>#/definitions/Property",
        "@value": 50
        },
        "https://schema.org/minValue": {
        "@type": "https://<IRI for core_defs.json>#/definitions/Property",
        "@value": -20
        },
        "https://schema.org/unitCode": {
        "@type": "https://<IRI for core_defs.json>#/definitions/Property",
        "@value": "CEL"
        },
        "https://schema.org/unitText": {
        "@type": "https://<IRI for core_defs.json>#/definitions/Property",
        "@value": "degree Celsius"
        }
  },
.
.
}
```

Note that ‘maxValue’ has expanded to ‘https://schema.org/maxValue’. This implies the property ‘unitCode’ referred to in the data schema is same as ‘schema.org/maxValue’. Applications already aware of that vocabulary will find it very easy to use and interpret this attribute. The above example also illustrates how IUDX catalogue is able to reuse attribute definitions from external vocabularies using “@context” keyword.

Another important aspect to note is that the “@context” field in data model also serves to provide context to the data attributes and hence may be used to convert the JSON data from the resource into JSON-LD data.

We next provide an example of the catalogue item of itemType ‘resourceItem’. This catalogue item summarizes various types of meta-information with regards to this data resource.

exItem.json

```
{
    "@context": [
        "<catalogue-link>/ex_c02Temp_sensor/ex_sensor_dataModel.json"
    ],
    "itemDescription": "An example catalogue resource item",
    "id": "urn:iudx-cat:abc/xyz/848f-4848-dxbe",
    "itemType": {
        "type": "Property",
        "value": "resourceItem"
    },
    "tags": {
        "type": "Property",
        "value": [ "environment", "air-quality", "climate", "air", "pollution", "CO2",
"Temperature"]
    },
    "refBaseSchema": {
        "value": "<catalogue-link>/resourceItem_schema.json",
        "type": "Relationship"
    },
    "refDataModel": {
        "value": "<catalogue-link>/ex_sensor_dataModel.json",
        "type": "Relationship"
    },
    "resourceId": {
        "type": "Property",
        "value": "CO2_Temp_Sensor"
    },
    "resourceServer": {
        "value": "urn:iudx-cat:636485-945454-2a9495",
        "type": "Relationship"
    },
    "resourceServerGroup": {
        "value": "urn:iudx-cat:abc357-gw2554-452abc",
        "type": "Relationship"
    },
    "provider": {
        "value": "urn:iudx-cat:ae48f2-5c01e8-3feda9",
        "type": "Relationship"
    },
    "NAME": {
        "type": "Property",
        "value": "CO2_Temp_Sensor"
    },
    "location": {
      "type": "GeoProperty",
      "value": {
          "geometry": {
            "type": "Point",
            "coordinates": [77.5703, 13.0138]
          }
      }
    },
    "accessInformation": [
        {
            "accessObjectType": "OpenAPI",
            "accessObject": {
                "value": "<catalogue-link>/exApiObject_c02Temp_sensor.json",
                "type": "Relationship"
            },
            "accessObjectVariables": {
                "type": "Property",
                "value": {
                    "resourceId": "CO2_Temp_Sensor"
                }
            }
        }
    ],
    "authorizationServerInfo": {
        "type": "Property",
        "value": {
            "authServer": "http://auth.iudx.org.in",
            "authType": "iudx-auth"
        }
    },
    "createdAt": {
        "type": "TimeProperty",
        "value": "2019-02-20T10:30:06.093121"
    },
    "itemStatus": {
        "type": "Property",
        "value": "active"
    }
}
```

The base schema for this item is specified by the field ‘refBaseSchema’ which in this case refers to the schema for an item of type ‘resourceItem’[15]. The above item contains all the mandatory attributes listed by the base schema. Similarly, ‘refDataModel’ points to the ‘data-model’ (described above). Note that links (attributes of type ‘Relationship’) are provided for the ‘`Provider`’ item (which contains information about ‘`Provider`‘ entity for this item) and ‘`resourceServer`’ item (which provides information about ‘`resource-server`’ entity on which this resource is hosted).

The item also contains ‘tags’ and ‘itemDescription’ attributes which are very useful for discovery purposes. Another important attribute of type ‘Relationship’ is the ‘accessObject’ which refers to the API object (described below) that describes methods for accessing data from this resource. The attribute ‘accessVariables’ lists down the values of API variables required by the API object. Using these attributes it becomes very easy for any consuming application to access data from this resource.

We also note that the item contains “@context” attribute which contains mappings for all the attributes in the item. Below we provide an illustrative snippet of the JSON-LD expanded version of the attribute ‘location’ contained in the above item.

exItem context .json

```
 ...
 "<catalogue-link>/airQuality/airQuality_context.json#/location": {
       "@type": "<catalogue-link>/core_defs.json#/definitions/GeoProperty",
       "<catalogue-link>/core_defs.json#/definitions/hasValue": {
       "https://purl.org/geojson/vocab#geometry": {
       "@type": "https://purl.org/geojson/vocab#Point",
       "https://purl.org/geojson/vocab#coordinates": {
```

[15] See https://github.com/iudx/iudx-ld

```
         "@list": [
         77.5703,
         13.0138
         ]
         }
         }
         }
   }


  ...
}
```

We once again see, from the above expansion, vocabulary reuse (in this case GeoJSON-LD) in IUDX catalogue framework.

Finally, we provide an example of ‘accessObject’. For this example, the data is assumed to be available through REST-APIs hosted on the ‘resource-server’ and hence the ‘accessObject’ can be an ‘openapi’ object as described below:

env_aqm_api.json

```
{
    "openapi": "3.0.1",
    "info": {
        "title": "TemperatureCO2Sensor",
        "version": "1.0.0"
    },
    "servers": [
        {
            "url": "https://iudx.resourceserver.org:8080/api/1.0.0/resource",
            "description": "Resource server"
        }
    ],
    "paths": {
        "/latest/aqm-temp-co2/{resourceId}": {
            "get": {
                "description": "Get the sensor message for a given resourceId",
                "responses": {
                    "200": {
                        "description": "Sensor Message as JSON",
                        "content": {
                            "application/json": {
                                "schema": {
                                    "$ref": "#/components/schemas/SensorPacket"
                                }
                            }
                        }
                    }
                }
            },
            "parameters": [
                {
                    "name": "resourceId",
                    "in": "path",
                    "description": "Id of the location where sensor is deployed",
                    "required": true,
                    "schema": {
                        "type": "string"
                    }
                }
            ]
        }
    },
    "components": {
        "schemas": {
            "SensorPacket": {
                "description": "JSON Packet response of the API",
                "type": "object",
                "properties": {
                    "NAME": {
                        "$ref":
"<catalogue-link>ex_c02Temp_sensor/ex_sensor_dataModel.json#/properties/NAME"
                    },
                    "LASTUPDATEDATETIME": {
                        "$ref":
"<catalogue-link>ex_c02Temp_sensor/ex_sensor_dataModel.json#/properties/LASTUPDATEDATETIME"
                    },
                    "CO2_MAX": {
                        "$ref":
"<catalogue-link>ex_c02Temp_sensor/ex_sensor_dataModel.json#/properties/CO2_MAX"
                    },
                    "TEMPERATURE_MAX": {
                        "$ref":
"<catalogue-link>ex_c02Temp_sensor/ex_sensor_dataModel.json#/properties/TEMPERATURE_MAX"
                    }
                }
            }
        }
    }
}
```

Note that the API object refers to the data model to describe attributes in its request-response bodies. The same API object variables, e.g. 'NAME' in the above example, pertaining to the individual resources should be provided via the 'accessVariables' attribute in the corresponding 'resourceItem'.

A message packet of a resource coming from a resource server might point to the context for that datamodel supporting dynamic semantic interpretability, for example -

example message packet (JSON)

`{`
`    "@context": "<catalogue-link>/airQuality/airQuality_dataModel.json",`

```
    "TEMPERATURE_MAX": 43,
    "CO2_MAX" : 980,
    "LASTUPDATETIME": "2019-08-14T06:10:39Z"
}
```

Including an "@context" field here which points to the previously mentioned airQuality_dataModel allows the receiver of this message to interpret the fields "TEMPERATURE_MAX", etc without knowing apriori the kind of resource that is sending this message.

Passing this message through a JSON-LD preprocessor yields

example message packet (JSON-LD)

```
{
  "<catalogue-link>/airQuality/airQuality_context.json#/CO2_MAX": {
    "@type": "<catalogue-link>/core_defs.json#/definitions/QuantitativeProperty",
    "@value": 980
  },
  "<catalogue-link>/airQuality/airQuality_context.json#/TEMPRATURE_MAX": {
    "@type": "<catalogue-link>/core_defs.json#/definitions/QuantitativeProperty",
    "@value": 43
  }
}
```

The id of an attribute, <catalogue-link>/airQuality/airQuality_context.json#/CO2_MAX provides for semantic interpretability whereby a user/program can trace the link and understand what exactly CO2_MAX means. The @type provides "type" interpretability where the structure of the attribute can be understood.